DECK36

Real-time Data De-duplication using Locality-sensitive Hashing powered by Storm and Riak

Dr. Stefan Schadwinkel
@ Berlin Buzzwords 2014

DECK36

DECK36

- DECK36 is an ICANS Spin-Off
- Small team of core engineers
- Longstanding expertise in complex web systems
- Developing own data intelligence-focussed tools and web services
- Offering our expert knowledge in:
  - Automation & Operations
  - Architecure & Engineering
  - Analytics & Data Logistics

Dr. Stefan Schadwinkel

Co-Founder / Analytics Engineer

stefan.schadwinkel@deck36.de

DUPLICATE DATA

Why is that interesting?

Databases

Sequence Tracking

Fraud

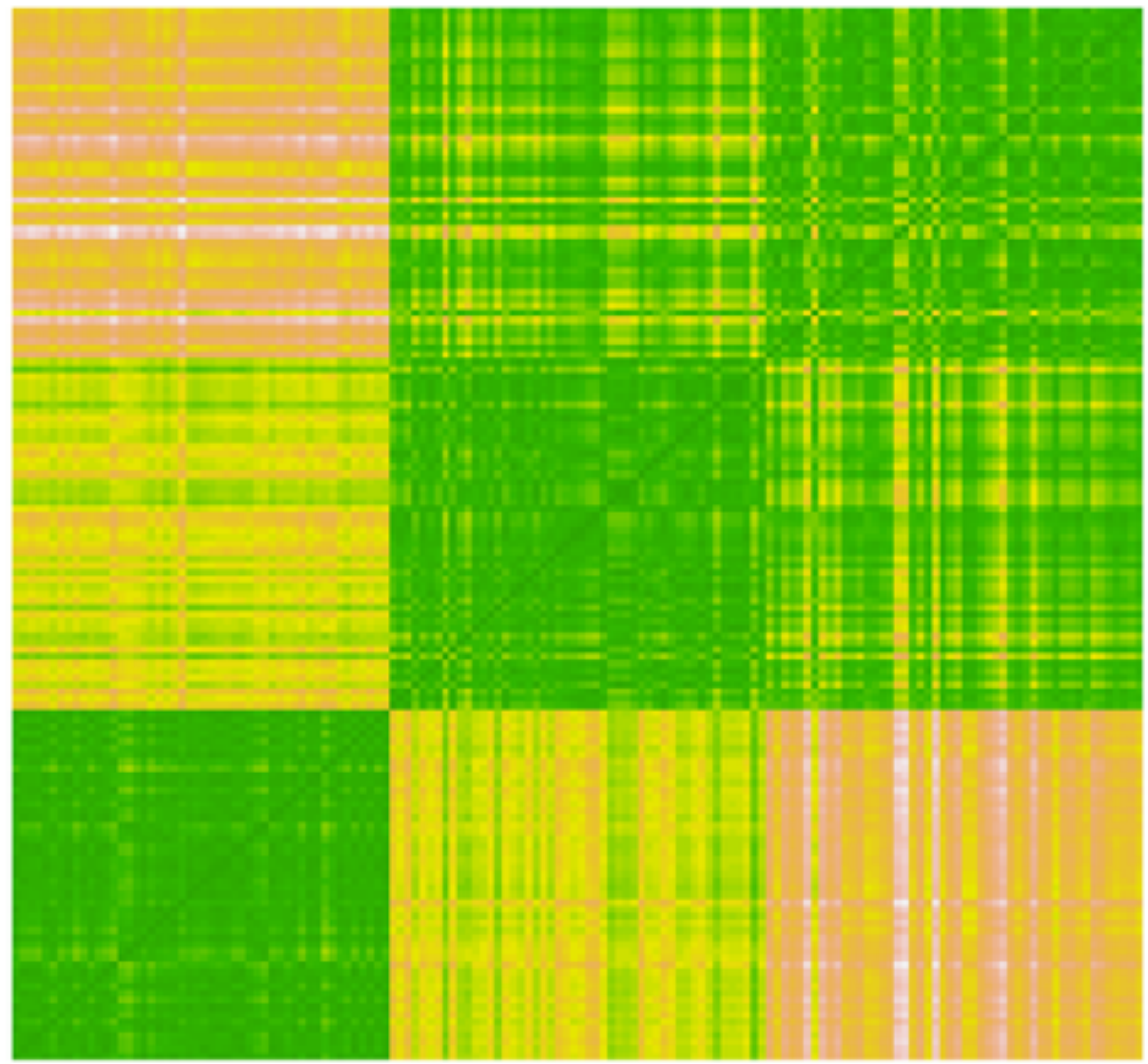
Fingerprinting

Clustering in general

Velocity

GENERAL CLUSTERING
The Mighty and Terrible …

… Distance Matrix.




matching
entity
RESOLUTION
IDENTIFICATION
DEDUPLICATION
record
DUPLICATE
DATA
IDENTITY
INTEGRATION
detection
OBJECT
HARDENING
PROCESSING
RECONCILIATION
INSTANCE
fuzzy
linkage
identification
MERGE
PURGE
WASHING
resoultion
COREFERENCE
DATABASE
LIST
INFORMATION
DISAMBIGUATION
LINKAGE
DETECTION
IDENTIFICATION
INTEGRATION
RESOLUTION
CONFLATION
HARDENING CONFLATION
DEDUPLICATION
LINKING
RECORD
processing
duplicate
identity
object
problem
reference
MATCHING
ENTITY
PURGE
FUZZY
disambiguation
database
list
data

DUPLICATE DATA
Some comments…

Record linkage (RL)

… refers to the task of finding records in a data set that refer to the ***same entity*** across ***different data sources*** (e.g., data files, books, websites, databases).

… joining data sets based on entities that may or ***may not share a common identifier***
… as may be the case ***due to differences in record shape, storage location, and/or curator style or preference.***

… ***records do not share a common key […] Errors*** are introduced as the result of ***transcription errors***, ***incomplete information***, ***lack of standard formats***, or any combination of these factors.

DUPLICATE DATA
Short recap.

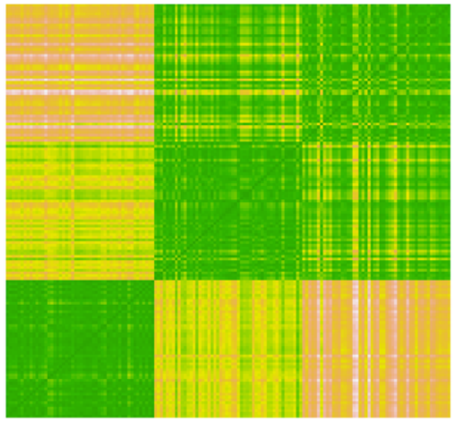
Elmagarmid et.al. (2007): "Duplicate Record Detection: A Survey"

- The probabilistic foundations: Dunn (1946) and Newcombe et.al. (1959).
- Fellegi and Sunter (1969) formalized the probabilistic foundations into what remains the mathematical foundation for many record linkage applications even today.
- Various machine learning techniques since the late 1990's.

Basic Method

- For each feature select a match-type (character-based, token-based, phonetic, ..)
- For each feature, define a weight (importance)
- Calculate total match value (sum of weight * match result)
- Use machine learning to learn the weights

DUPLICATE DATA
Grrr, that terrible daemon!



Captain Obvious: It’s prohibitively expensive.

- Single step divide and conquer called “Blocking”
- Group data into “blocks” and perform the comparisons only within the blocks
- Multi-pass approaches

It’s artisan craftwork.

DISCUSSION
Meet Dedoop

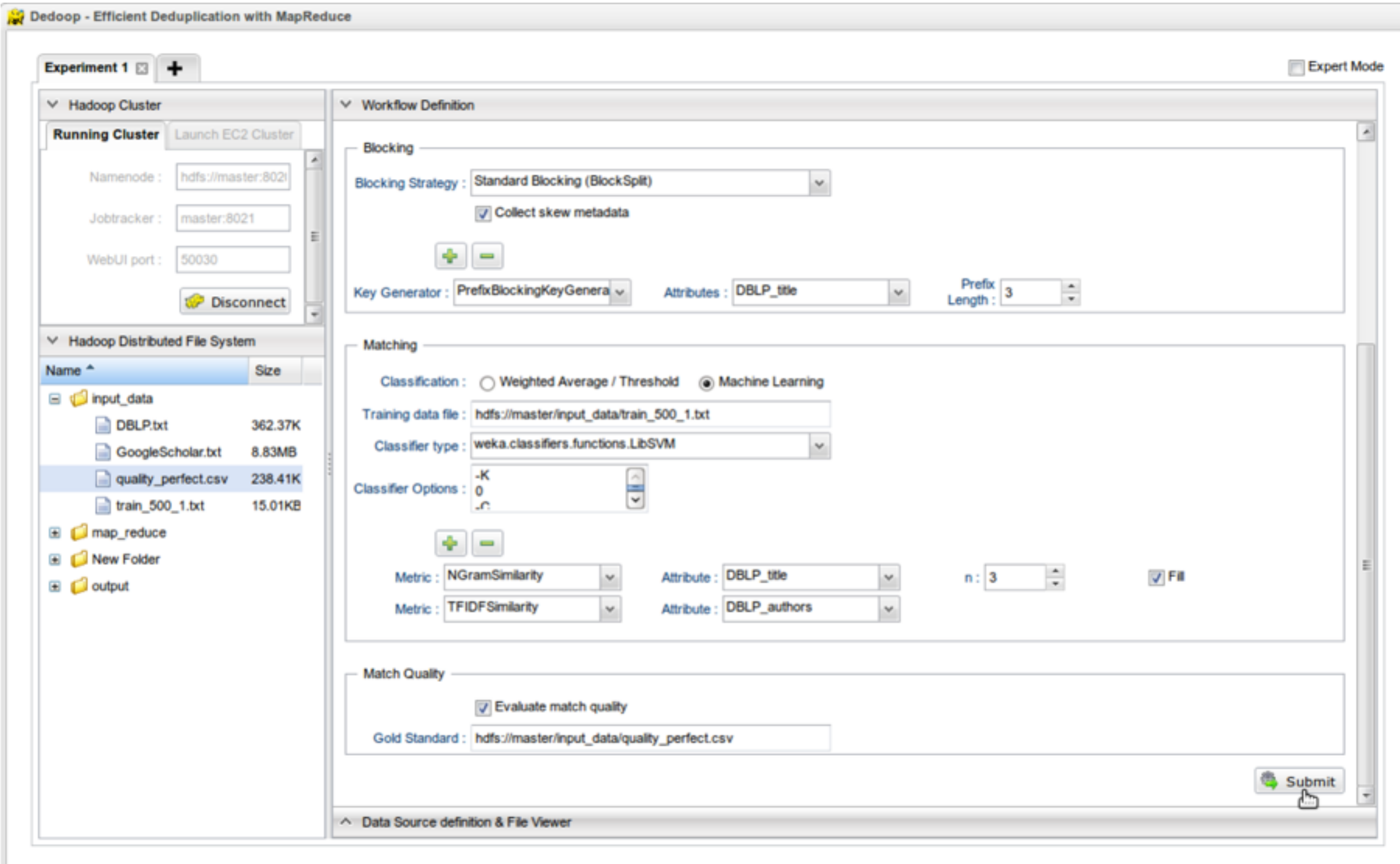
Dedoop: It’s an environment for Deduplication Craftmanship

- It’s a research project from Leipzig University
- http://dbs.uni-leipzig.de/research/projects/large_scale_object_matching

Research Focus Topics

- Skew handling / Load balancing
  - Because, you know, some blocks are bigger than others …
- Redundancy-free comparisons
  - Multi-pass approaches lead to overlapping blocks
  - Overlapping blocks lead to comparisons you already made…

DISCUSSION
Meet Dedoop



MULTI-INDEX LSH

The Algorithm

MULTI-INDEX LSH
Overview

Not my invention.

The method is from the Eventbrite Engineering Blog.

Kudos to Jay Chan! And all the other guys.

https://engineering.eventbrite.com/multi-index-locality-sensitive-hashing-for-fun-and-profit/

Main points:

1. The entity representation is transformed using the MinHash algorithm.
2. The number of comparisons is reduced using an indexing scheme.

MULTI-INDEX LSH
Tokenization

Free Text

A: “The world is for you to get over with!”
➔ the, world, is, for, you, to, get, over, with

B: “Let’s take over the world!”
➔ lets, take, over, the, world

Jaccard Similarity Coefficient

Union Set of Tokens: {the, world, is, for, you, to, get, over, with, lets, take}
Intersection of Tokensets: {the, world, over}
JSC = size of intersection / size of union = 3 / 11 ≈ 27%

MULTI-INDEX LSH
Jaccard Similarity by MinHash

Just count how many hashes match.

Approx. Jaccard = # of matching hashes / # hashes

A: Bag of 32 hashes

B: Bag of 32 hashes

Let’s say, 10 hashes are identical,
then the Jaccard Similarity is approximately 10 / 32
which is ~ 0.31

MULTI-INDEX LSH
MinHash

“hash each token. take the minimum. repeat N times.”

Transforms a bag of tokens to a bag of N hash values (32bit integers)

```java
// apply each hasher
for (int i = 0; i < n; i++) {

    HashFunction localHasher = nHashers.get(i);

    // Initialize the minimum hash with
    // the *unsigned* MAX
    Integer minHash = Integer.MIN_VALUE;

    // apply this hasher to each token
    for (String token : tokenBag) {

        int kHash = localHasher.hashString(token).asInt();

        if (unsignedLowerThan(kHash, minHash)) {
            minHash = kHash;
        }
    }

    hashes.add(i, minHash);
}
```

MULTI-INDEX LSH
Bit Sampling

Minhashing so far reduced the price per comparison, but we would still need to compare all hash bags with all hash bags. But first, let's reduce the price further.

1. We don't need to know the exact hash, we only want to know if they match.
2. If one bit does not match, we know the two hashes can't be identical.

➔ just keep the least significant bit. 32 hash values become one single 32 bit number.

Of course the LSB will be identical in 50% of the cases at random, but we can adjust for that.

MULTI-INDEX LSH
Multi-Index “Blocking”

Norouzi et.al. (2012) “Fast Search in Hamming Space with Multi-Index Hashing"

http://www.cs.toronto.edu/~norouzi/research/mih/

https://github.com/norouzi/mih

Based on the “Pidgeonhole Principle”

If I put 10 items into 9 pigeonholes…

- then there must be at least one, that has >= 2 items.
- then there must be at least one, that has 0 or 1 item.

If I put N item into M containers…

- then there must be at least one, that has >= ceil(N/M) items [ceil(10/9) = 2]
- then there must be at least one, that has <= floor(N/M) items [floor(10/9) = 1]

MULTI-INDEX LSH
Multi-Index “Blocking”

We can use that principle to build an index for our hash bitsamples.

32 hash functions ➔ One 32 bit sample

90% Match ➔ 28 bits must match ➔ up to 4 can be unequal (➔ *N*)

Split the 32 bit sample in 4 chunks of 8 bit ➔ at least one chunk has 0 or 1 unequal bit

These chunks will now become our index. Imagine a map:

1st chunk ➔ [full 32 bit, …]

2nd chunk ➔ [full 32 bit, …]

3rd chunk ➔ [full 32 bit, …]

4th chunk ➔ [full 32 bit, …]

As you see, we trade space for time…

MULTI-INDEX LSH
Multi-Index “Blocking”

Our candidates are now those in the list behind the matching chunk.

Split the 32 bit sample in 4 chunks of 8 bit ➔ at least one chunk has 0 or 1 unequal bit

➔ All relevant candidates are behind those chunks

Lookup:

– Split bit sample into chunks

– Take all candidates behind <u>one</u> chunk and its bit distance siblings

– You can stop, once you have candidates based on one source chunk

– With all the candidates you then perform the final comparisons

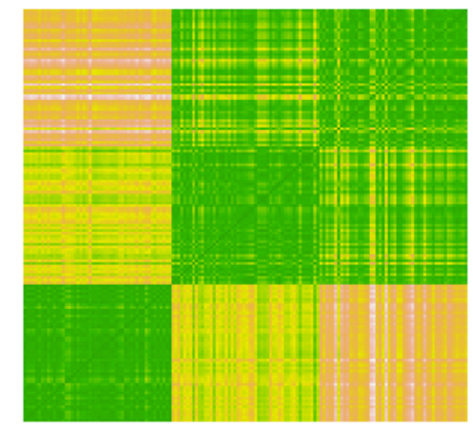
MULTI-INDEX LSH
Multi-Index “Blocking”

Performance boundaries

M - total number of messages to compare

N - number of hash functions resp. bits in the sample

K - number of chunks the bit sample is split into



Insert

O(1) - compute the K chunks and store in map

Lookup

O(1) - compute the K chunks and lookup the candidate set

$O(M * M / 2^{(N/K)})$ - compare every message (M) with all candidates ($M / 2^{(N/K)}$)

If we choose N and K so that $\mathbf{2^{(N/K)} >> M}$, we can now achieve O(M), i.e **linear scaling.**

HANDS ON!

Storm and Riak.

HANDS ON!
Mail Grouping for Spam Detection

I wanted something else, but: Available data!

http://untroubled.org/spam/

“This directory contains all the spam that I have received since early 1998. I have employed various "bait" addresses, … to trick email address harvesters into putting them on spam lists.”

Spam mails from 2014

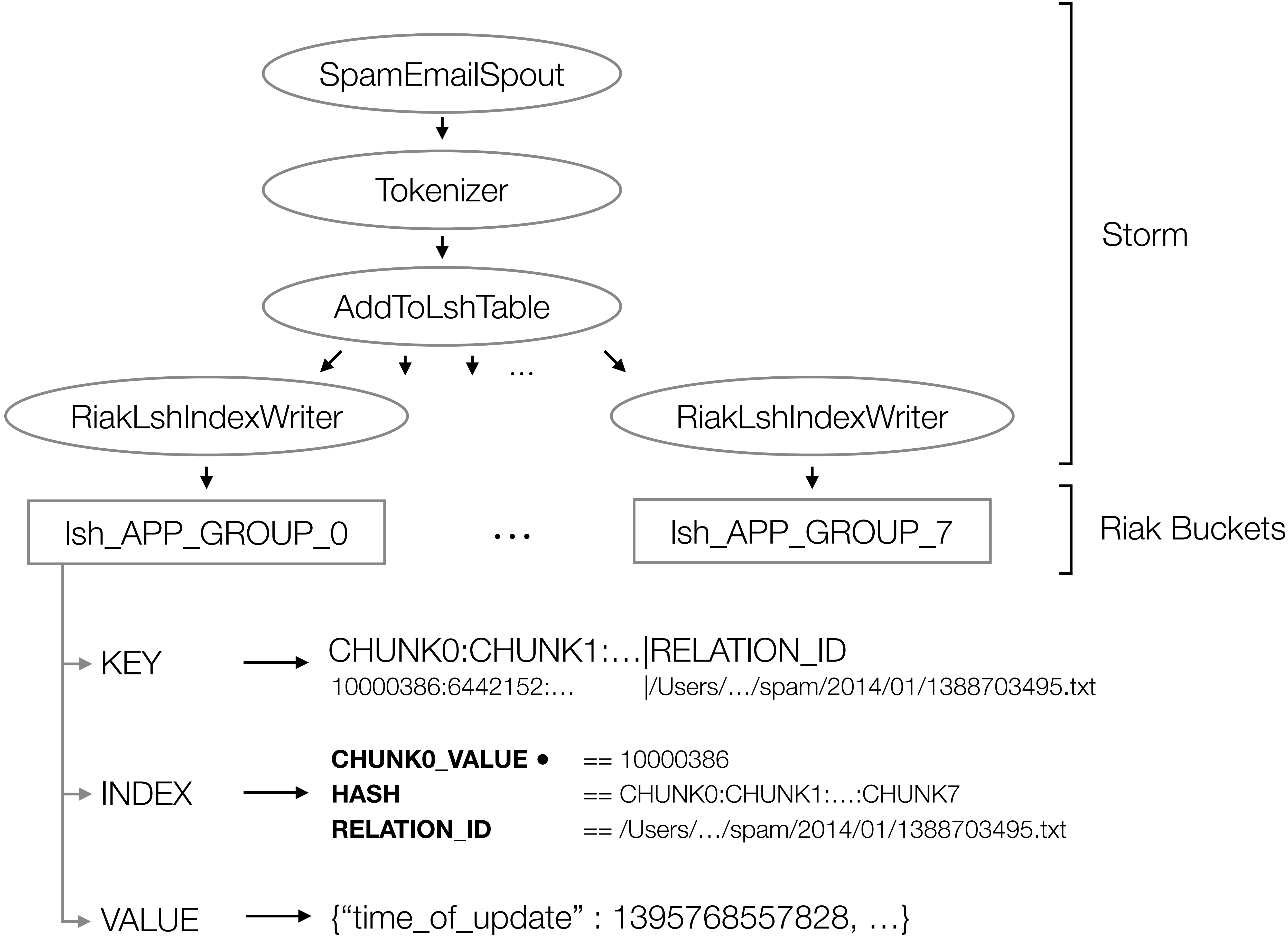
January + February: 85500 Mails

$2^{(N/K)}$ >> M

2^{24} ~ 16 million messages

K - 8 chunks à 24 bit

N - 192 hash functions


SpamEmailSpout
Tokenizer
AddToLshTable
RiakLshIndexWriter
RiakLshIndexWriter
Storm
lsh_APP_GROUP_0
lsh_APP_GROUP_7
Riak Buckets
KEY
CHUNK0:CHUNK1:…|RELATION_ID
10000386:6442152:… |/Users/…/spam/2014/01/1388703495.txt
INDEX
CHUNK0_VALUE •
== 10000386
HASH
== CHUNK0:CHUNK1:…:CHUNK7
RELATION_ID
== /Users/…/spam/2014/01/1388703495.txt
VALUE
{"time_of_update" : 1395768557828, …}

DEMO

THE FUTURE

Some ideas.

THE FUTURE
Applications

Fingerprinting

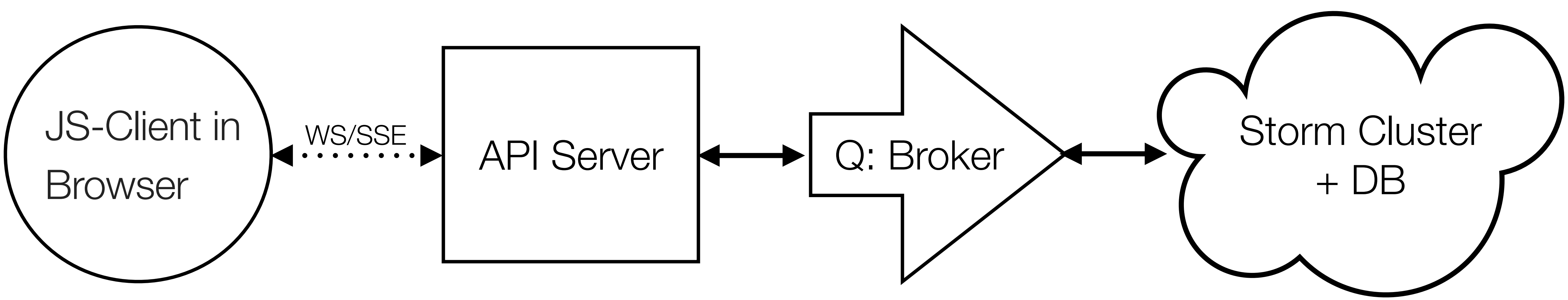
Create tokens from data available with JavaScript and HTTP Request:

A: {“browser”:”Mozilla Firefox”,”plugins”:”Flash,Silverlight”,”zipcode”:20121}

➔ browser_mozilla, browser_firefox, plugin_flash, plugin_silverlight, zipcode_20121

B: {“browser”:”Google Chrome”,”plugins”:”Flash,Java”,”likes”:[123,124]}

➔ browser_google, browser_chrome, plugin_flash, plugin_java, like_123, like_124



THE FUTURE
Applications

Sequence Tracking

Sequence Code: step1_$URL, step2_$URL, …

Set Code: $URL1, $URL2, …

- Gets you the most common ‘fuzzy’ sequences or sets
- Variation: use a window for X successively visited sites, products, etc.
- Add features, i.e. sets of bought items

THE FUTURE
Applications

Fraud

Often identities are generated after some pattern with variations:

A: a_lastname32@yahoo.com
B: alex_lastname746@yahoo.co.uk

- Generally use redundant encoding using multiple encodings
- Remove whitespace, split & use q-grams
- Use phonetic encoding schemes
- Use special knowledge about email, address, etc. to create tags
- Use pre-filters and artisan blocking (i.e. major free mail providers)

– Low number of tokens
– Needs some research

Thank You.