Efficient Online Locality Sensitive Hashing via Reservoir Counting

Benjamin Van Durme and Ashwin Lall
Locality Sensitive Hashing

Hamming Distance := \( h = 1 \)
Signature Length := \( b = 6 \)

\[
\cos(\theta) \approx \cos\left(\frac{h}{b} \pi\right) = \cos\left(\frac{1}{6} \pi\right)
\]
Accuracy as function of bits

![Accuracy plots](image)

- **32 bit signatures**
  - Approximate Cosine
  - True Cosine
  - Cheap

- **256 bit signatures**
  - Approximate Cosine
  - True Cosine
  - Accurate
Some uses in Comp Ling

Noun Clustering
  Ravichandran, Hovy and Pantel (2005)

Paraphrase Acquisition
  Bhagat and Ravichandran (2008)
  Chan, Callison-Burch and Van Durme (2011)

Lexicon Induction
  Bergsma and Van Durme (2011)

Topic Detection and Tracking (TDT)
  Petrovic, Osborne and Lavrenko (2010)
LSH

1 2 3 ... b
LSH

\[ \vec{v} \cdot \vec{r}_i \geq 0 \]

\[ \vec{v} \cdot \vec{r}_i < 0 \]
Online LSH

\[ \sum_t \vec{v}_t \cdot \vec{r}_i \geq 0 \]

\[ \sum_t \vec{v}_t \cdot \vec{r}_i < 0 \]
Online LSH : $b$ parallel streams of numbers

\[ 3 + -2 + 4 + 1 + -1 + \ldots < 0 \]
\[ -5 + -1 + 7 + -2 + 3 + \ldots \geq 0 \]
\[ \ldots \]
\[ 1 + 2 + -4 + 5 + -2 + \ldots < 0 \]
Stream of numbers

3, -2, 4, 1, -1, ...
Stream of numbers

\[
\begin{array}{c|l}
 n &= 0 \\
 s &= 0 \\
 3, -2, 4, 1, -1, & \ldots
\end{array}
\]
Stream of numbers

position in stream

\[ n=0 \]
\[ s=0 \]
\[ 3, -2, 4, 1, -1, \ldots \]
Stream of numbers

\[
\begin{align*}
n &= 0 \\
3, -2, 4, 1, -1, \ldots
\end{align*}
\]

\[
\begin{align*}
s &= 0
\end{align*}
\]

sum up until \( n \)
Stream of numbers

n=0
3, -2, 4, 1, -1, ...

s=0
Stream of numbers

n=1
3, -2, 4, 1, -1, ...
s=3
Stream of numbers

n=2
3, -2, 4, 1, -1, ...
s=1
Stream of numbers

$n=3$

$3, -2, 4, 1, -1, ...$

$s=5$
Stream of numbers

\[ n = 4 \]
\[ 3, -2, 4, 1, -1, \ldots \]
\[ s = 6 \]
Stream of numbers

\[ n=5 \]

3, -2, 4, 1, -1, ...

\[ s=5 \]
Size of variables

\[ n \]
\[ \ldots, \ldots, \ldots, \ldots, \ldots, \ldots \]
\[ s \]
Size of variables

(int32) n

..., ..., ..., ..., ..., ...

(int32) s
Size of variables

(int32) n
...
(int8) s

make this smaller via approximation
Reservoir Sampling

a, b, c, d, e, ...
Reservoir Sampling

\[ a, b, c, d, e, \ldots \]

reservoir of size \( k=3 \)

\[ [?, ?, ?] \]
Reservoir Sampling

\[ n=0 \]

\[ a, b, c, d, e, \ldots \]

\[ [?, ?, ?, ?] \]
Reservoir Sampling

\[ n=1 \]

\[ a, b, c, d, e, \ldots \]

\[ [a, ?, ?] \]
Reservoir Sampling

\[ n=2 \]

\[ a, b, c, d, e, \ldots \]

\[ [a, b, ?] \]
Reservoir Sampling

n=3

\[ a, b, c, d, e, \ldots \]

\[ [a, b, c] \]
Reservoir Sampling

\[ n=4 \]

\[ a, b, c, d, e, \ldots \]

?
Reservoir Sampling

\[ n=4 \]
\[ a, b, c, d, e, \ldots \]
\[ ? \]

1. accept \textbf{d} with probability: \( k/n = 3/4 \)
Reservoir Sampling

n=4

a, b, c, d, e, ...

?  

1. accept d with probability: $k/n = 3/4$
2. if accept, then **evict** a random element from reservoir
Reservoir Sampling

n=4

a, b, c, d, e, ...

[a, b, c]

1. accept d with probability: k/n = 3/4
2. if accept, then evict a random element from reservoir
Reservoir Sampling

n=4

a, b, c, d, e, ...

[a, b, c]

1. accept d with probability: k/n = 3/4
2. if accept, then evict a random element from reservoir
Reservoir Sampling

n=4

a, b, c, d, e, ...

[a, d, c]

1. accept d with probability: $k/n = 3/4$
2. if accept, then **evict** a random element from reservoir
Reservoir Sampling

$n=5$

$a, b, c, d, e, ...$

$[a, d, c]$

1. accept e with probability: $k/n = 3/5$
Reservoir Sampling

n=26

a, b, c, d, e, ...

[p, h, z]
Stream of numbers

3, -2, 4, 1, -1, ...
Stream of numbers in +/- unary

3, -2, 4, 1, -1, ...

1, 1, 1
Stream of numbers in +/- unary

3, -2, 4, 1, -1, ...

1, 1, 1, -1, -1
Stream of numbers in +/- unary

3, -2, 4, 1, -1, ...

1, 1, 1, -1, -1, 1, 1, 1, 1
Stream of numbers in +/- unary

3, -2, 4, 1, -1, ...

1, 1, 1, -1, -1, 1, 1, 1, 1, 1
Stream of numbers in +/- unary

3, -2, 4, 1, -1, ...
1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...
Stream of numbers in +/- unary

1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...
Stream of numbers in +/- unary

\[
\begin{align*}
n &= 0 \\
1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots
\end{align*}
\]

sum up until \( n \)
Stream of numbers in +/- unary

n=1
1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...

s=1
Stream of numbers in +/- unary

\[ n=2 \]
\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1, -1, \ldots \]
\[ s=2 \]
Stream of numbers in +/- unary

\[ ... \]

\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ... \]
Stream of numbers in +/- unary

\[ n=6 \]

1, 1, 1, -1, -1, 1, 1, 1, 1, -1, ...

\[ s=2 \]
Stream of numbers in +/- unary

1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...
Stream of numbers in +/- unary

\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]

\[ n=11, \quad s=5 \]
Sampling from stream of numbers in +/- unary

1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...
Sampling from stream of numbers in +/- unary

\[ n=0 \]
\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ... \]

[?,?,?]
Sampling from stream of numbers in +/- unary

\[ n=3 \]

\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1, -1, \ldots \]

\[ [1, 1, 1] \]
Sampling from stream of numbers in +/- unary

\[ n=5,\ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]

\[ [1,-1,1] \]
Sampling from stream of numbers in +/- unary

\[ [1, 1, 1] \]

\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ... \]

\( n=9 \)
Exchangeability of elements

\[ n=5 \]

\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]

\[ [1,-1,1] \]
Exchangeability of elements

\[
\begin{align*}
n &= 5 \\
& \quad \begin{bmatrix} 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \end{bmatrix} \\
& \quad [1,-1,1] \\
& \quad [-1,1,1] \\
& \quad [1,1,-1]
\end{align*}
\]
Implicit reservoir

n=5

1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...

“two 1’s, one -1”

[1,-1,1]
[-1,1,1]
[1,1,-1]
Implicit reservoir

"two 1's, one -1"
"two 1's, k=3"

\[\begin{array}{c}
1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \\
[1,-1,1] \\
[-1,1,1] \\
[1,1,-1]
\end{array}\]
Implicit reservoir

If $k=3$, there can be:
zero 1’s,
one 1,
two 1’s, or
three 1’s.
Implicit reservoir

If k=3, there can be:
zero 1’s,
one 1,
two 1’s, or
three 1’s.

For a given k, there can be:
k+1 different states,
representable with lg(k+1) bits.
Implicit reservoir

If \( k=3 \), there can be:
- zero 1’s,
- one 1,
- two 1’s, or
- three 1’s.

For a given \( k \), there can be:
- \( k+1 \) different states,
- representable with \( \log(k+1) \) bits.

When \( k=255 \), need \( \log(256) = 8 \) bits.
Sampling from stream of numbers in +/- unary

\[ n=0 \]
\[ s=0 \]
\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]

[?, ?, ?]
Sampling from stream of numbers in +/- unary

\[ n=0 \]
\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]
\[ s=0 \]
\[ [\_, \_, \_?] \]

sum of 1’s in the reservoir
Sampling from stream of numbers in +/- unary

\[ n=3 \]
\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]
\[ s=3 \]
\[ [1,1,1] \]
Sampling from stream of numbers in +/- unary

n=5
1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...
s=2

[-1,1,1]
[1,-1,1]
[1,1,-1]
Sampling from stream of numbers in +/- unary

\[ n=9 \]

\[ 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots \]

\[ s=3 \]

\[ [1,1,1] \]
Sampling from stream of numbers

3, -2, 4, 1, -1, ...

1, 1, 1, -1, -1, 1, 1, 1, 1, -1, ...
Sampling from stream of numbers

3, -2, 4, 1, -1, ...

n=5

1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...

s=2
Sampling from stream of numbers

Compute **expected** number of accepts, then **expected** change to reservoir.

\[3, -2, 4, 1, -1, ...\]

\[n=5\]

\[1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...\]

\[s=2\]
Sampling from stream of numbers

Compute expected number of accepts, \( k \log((n+m)/n) \) then expected change to reservoir.

\[
3, -2, 4, 1, -1, \ldots
\]

\[
n=5
\]

\[
1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, \ldots
\]

\[
s=2
\]
Sampling from stream of numbers

Compute expected number of accepts, then expected change to reservoir. [see paper]

3, -2, 4, 1, -1, ...

n=5

1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, ...

s=2
Online LSH : b parallel streams

3, -2, 4, 1, -1, ...

-5, -1, 7, -2, 3, ...

...
Online LSH : b parallel streams

n≈10
3, -2, 4, 1, -1, ...
\(s_1=2\)
-5, -1, 7, -2, 3, ...
\(s_2=0\)
...
...
1, 2, -4, 5, -2, ...
\(s_b=1\)
Online LSH : b parallel streams

from int32 to \lg(k+1) bits, e.g., int8

\[ n \approx 10 \]
\[ s_1 = 2 \]
\[ s_2 = 0 \]
\[ s_b = 1 \]
More bang for the bit

![Graph showing Mean Absolute Error vs. Bits Required](image-url)

- **log2.k**
  - 8
  - 32

- **b**
  - 64
  - 128
  - 192
  - 256
  - 512

ACL 2011

Benjamin Van Durme & Ashwin Lall
More bang for the bit

Reservoir Counting

log2.k
- 8
- 32

b
- 64
- 128
- 192
- 256
- 512

Mean.Absolute.Error

Bits.Required

- 1000
- 2000
- 3000
- 4000
- 5000
- 6000
- 7000
- 8000

log2.k

b
More bang for the bit

Explicit Counting

Mean.Absolute.Error

Bits.Required

log2.k

- 8
- 32

b

- 64
- 128
- 192
- 256
- 512

More bang for the bit
More bang for the bit

![Graph showing mean absolute error against bits required for different values of log2(k) and b. The graph shows a decreasing trend as the number of bits increases.](image)
More bang for the bit

50% memory, same error
More bang for the bit

Same memory, less error
Thanks