Computing the Entropy of a Stream

To appear in SODA 2007

Graham Cormode

graham@research.att.com

Amit Chakrabarti

Dartmouth College

Andrew McGregor

U. Penn / UCSD

Outline

- Introduction
- Entropy Upper Bound
- Lower Bounds and Higher Orders
- Random Walks on Graphs

Bertinoro

McGregor

Chakrabarti

Entropy

A simple problem...?

- Given a long sequence of characters

$$S = \langle a_1, a_2, a_3 \ldots a_m \rangle \qquad \text{each } a_j \in \{1 \ldots n\}$$

- Let f_i = frequency of i in the sequence
- Compute the empirical entropy:

$$H(S) = - \sum_i f_i/m \log f_i/m = - \sum_i p_i \log p_i$$

- Example: S = < **`a, b, a, b, c, a, d, a`**>
  - $p_a = 1/2$, $p_b = 1/4$, $p_c = 1/8$, $p_d = 1/8$
  - H(S) = ½ + 1/4.2 + 1/8.3 + 1/8.3 = 7/4

Challenge

- Goal: approximate $H(S)$ in space sublinear (poly-log) in m (stream length), n (alphabet size)
  - (ε,δ) approx: answer is $(1\pm\varepsilon)H(S)$ w/prob $1-\delta$
- Easy if we have $O(n)$ space: compute each f_i exactly
- More challenging if n is huge, m is huge, and we have only one pass over the input in order
  - (The data stream model)

Motivation

- Entropy promoted for anomaly detection in networks
- If entropy (surrogate for distribution) suddenly changes, suspect anomaly
- More subtle than "heavy hitters" approach
- Approaches to computing entropy so far
  - Exact exhaustive computation
  - Heuristic: using compression size as surrogate for entropy
- In general, dimensionality and data are large (2^{32}, 2^{64}…)

Prior Work

- Guha, McGregor, Venkatasubramanian 2006
  - Gave $O(\mathbf{1/H(S)}1/\varepsilon^2 \log 1/\delta)$ space algorithm, amongst others
- Chakrabarti, Do Ba, Muthukrishnan 2006
  - Gave $O(m^{2\alpha}\ \text{polylog})$ space for $1/\alpha$ approximation
  - (ε,δ) approximation in $O(m^{2/3}\ \text{polylog})$
- Lall, Sekar, Ogihara, Xu, Zhang 2006
  - Partially heuristic approach, estimating a related quantity
- Bhuvanagiri, Ganguly 2006
  - $O(1/\varepsilon^3 \log^5 m)$ space algorithm (allows "deletions")

Basic Idea (via AMS)

- Simple estimator:
  - Randomly sample a position j in the stream
  - Count how many times a_j appears subsequently = r
  - Output $X = -(r \log r/m - (r-1) \log(r-1)/m)$

- Claim: $E[X] = H(S)$
  - Proof: prob of picking $j = 1/m$, sum telescopes correctly
- $\mathrm{Var}[X] = O(\log^2 m)$
  - Can be proven by bounding $|X| \leq \log m$

Analysis of Basic Estimator

- To get a good estimate, try to apply Chebyshev bounds
- Depends critically on ratio $Var[X]/E^2[X] = O(\log^2 m/H^2(S))$
- Problem: what happens when $H(S)$ is very small?
- Space needed for an accurate approx goes as $1/H^2$!

Low Entropy

- But... what does a low entropy stream look like?
  - **aaaaaaaaaaaaaaaaaaaaaaaaaaaaabaaaaa**
- Very boring most of the time, we are only rarely surprised
- Can there be two frequent items?
  - **aabababababababaababababbababababababa**
  - No! That's high entropy ($\approx$ 1 bit / character)
- Only way to get $H(S) = o(1)$ is to have only one character with p_i close to 1

Removing the boring guy

- Write entropy as
  - $-p_a \log p_a + (1-p_a) H(S')$
  - Where S' = stream S with all '**a**'s removed
- Can show:
  - Doesn't matter if $H(S')$ is small: as p_a is large, additive error on $H(S')$ ensures relative error on $(1-p_a)H(S')$
  - Relative error $(1-p_a)$ on p_a gives relative error on $p_a \log p_a$
  - Summing both (positive) terms gives relative error overall

Finding the boring guy

- Ejecting **a** is easy if we know in advance what it is
  - Can then compute p_a exactly
- Can find online deterministically
  - Assume $p_a > 2/3$ (if not, $H(S) > 0.9$, and original alg works)
  - Run a ‘heavy hitters’ algorithm on the stream
  - Modify analysis, find **a** and $p_a \pm \varepsilon\,(1-p_a)$
- But... how to also compute $H(S')$ simultaneously if we don’t know **a** from the start... do we need two passes?

Always have a back up plan...

- Idea: keep two samples to build our estimator
  - If at the end one of our samples is ‘`a`’, use the other
  - How to do this and ensure uniform sampling?
- Base on ‘min-wise sampling’:
  - For each token in the stream, pick a random label in the range [0...1]
  - Keep the token which has the smallest label
  - Each token has uniform probability of being picked

Sampling One Token

Stream:	C	A	A	B	B	A	B	D	C	A	B	A
Tags:	0.408	0.815	0.217	0.191	0.770	0.082	0.366	0.228	0.549	0.173	0.627	0.202
Repeats:						A				A		A

min tag

- Assign random tag ∈ [0,1] for each token
- Choose token with min tag (= uniform random choice)
- Implementation: keep track of
  (min tag, corresponding token, number of repeats)

Back up sampling

- If at the end of the stream the sampled character = ‘**a**’, we want to sample from the stream ignoring all ‘**a**’s
- This is just “the character achieving the smallest label distinct from the one that achieves the smallest label”
- Can track information to do this in a single pass, constant space

Sampling Two Tokens

Stream:	C	A	A	B	B	A	B	D	C	A	B	A
Tags:	0.408	0.815	0.217	0.191	0.770	0.082	0.366	0.228	0.549	0.173	0.627	0.202
Repeats:				B	B	A	B			A	B	A

min tag amongst remaining tokens

min tag

second smallest tag, but we don't want this; same token as min tag!

- Assign tags, choose first token as before
- Delete all occurrences of first token
- Choose token with min remaining tag; count repeats
- Implementation: keep track of two triples
  (min tag, corresponding token, number of repeats)

Putting it all together

- Can combine all these pieces
- Build an estimator based on tracking this information, deciding whether there is a boring guy or not
- A slightly fiddly Chernoff bounds argument improves number of repetitions of estimator from $O(\varepsilon^{-2}\mathrm{Var}[X]/E^2[X])$ to $O(\varepsilon^{-2}\mathrm{Range}[X]/E[X]) = O(\varepsilon^{-2} \log m)$

- In space $O(\varepsilon^{-2} \log m \log 1/\delta)$ space we can compute an (ε,δ) approximation to $H(S)$ in a single pass

Sliding Window Computation

- Suppose we only want entropy of last W tokens
- Observe we want to find min label so its token in last W
- Can find, if current minimum not in range, what would be next smallest token?
  - Expect smallest token to be $\approx W/2$ ago
  - Next smallest $\approx W/4$, then $W/8$…
  - Whp., need to keep $\log W$ candidates
- Extend analysis to tracking minimum and backup: need $\log^2 W$ with high probability
- Also need to find p_a in window with sufficient accuracy

Outline

- Introduction
- Entropy Upper Bound
- Lower Bounds and Higher Orders
- Random Walks on Graphs

Lower Bound

GAP−HAMM communication problem:

- Alice holds $x \in \{0,1\}^N$, Bob holds $y \in \{0,1\}^N$
- Promise: $\Delta(x,y)$ is either $\leq N/2$ or $\geq N/2 + \sqrt{N}$
- Which is the case?
- Model: one message from Alice to Bob

Requires $\Omega(N)$ bits of communication

[Indyk, Woodruff'03, Woodruff'04]

Lower Bound, Reduction

Alice: $x \in \{0,1\}^N$, Bob: $y \in \{0,1\}^N$

Entropy estimation algorithm **A**

- Alice runs **A** on $\mathrm{enc}(x) = \langle (1,x_1), (2,x_2), \ldots, (N,x_N) \rangle$
- Alice sends over memory contents to Bob
- Bob continues **A** on $\mathrm{enc}(y) = \langle (1,y_1), (2,y_2), \ldots, (N,y_N) \rangle$

Alice	0	1	0	0	1	1
	(1,0)	(2,1)	(3,0)	(4,0)	(5,1)	(6,1)
	(1,1)	(2,1)	(3,0)	(4,0)	(5,1)	(6,0)
Bob	1	1	0	0	1	0

Lower Bound

- Observe: there are
  - $2\Delta(x,y)$ tokens with frequency 1 each
  - $N-\Delta(x,y)$ tokens with frequency 2 each
- So, $H(S) = \log N + \Delta(x,y)/N$
- Thus size of Alice's memory contents = $\Omega(N)$.
  Set $\varepsilon = 1/(\sqrt{(N)} \log N)$ to show bound of $\Omega(\varepsilon/\log 1/\varepsilon)^{-2})$

Alice	0	1	0	0	1	1
	(1,0)	(2,1)	(3,0)	(4,0)	(5,1)	(6,1)
	(1,1)	(2,1)	(3,0)	(4,0)	(5,1)	(6,0)
Bob	1	1	0	0	1	0

Higher Orders

- Define $f_{i0i1\ldots it}$ = frequency of substring $i_0, i_1 .. i_t$
- Define $p_{it|i0i1\ldots it-1} = f_{i0i1\ldots it}/f_{i0i1\ldots it-1}$
- $H_k(S) = -\sum_{i0} p_{i0} \sum_{i1} p_{i1|i0} \ldots \sum_{ik} p_{ik|i0i1\ldots ik-1} \log p_{ik|i0\ldots ik-1}$

- Reduce $H_1(S)$ to **`PREFIX`** problem:
  - Alice has bitstring $x \in \{0,1\}^N$, Bob has bitstring $y \in \{0,1\}^M$
  - Bob determines if y is a prefix of x
  - Show communication complexity of **`PREFIX`** is $\Omega(N/\log N)$ – else Bob could determine x one bit at a time

Reduction to PREFIX

- Same encoding: enc(101)→ ⟨`(1,1) (2,0) (3,1)`⟩
- For $H_1(S)$, observe that if y is a prefix of x then the stream enc(y)enc(x) has every "character" `(i,b)` followed by the same `(j,c)`, so $H_1(S)=0$
- Else it is non-zero, so approximation could distinguish
- Thus, cannot approximate with o(m/log m) space.

(Partly a defect of the definition – a little unnatural that such a long string holds "zero" information)

Positive Result

- Can additively approximate $H_k(S)$:
- Write $H_k(S) = H(S^{k+1}) – H(S^k)$
  – Where $S^k = S$ with a new token for each k substring
  – E.g. $S =$ **`101011`** $S^2 =$ **`2 1 2 1 3`**
- Relative error approximate each term up to $\varepsilon/(2k \lg n)$ – since $H(S^k) \leq k \log n$, error is $\pm\, \varepsilon$

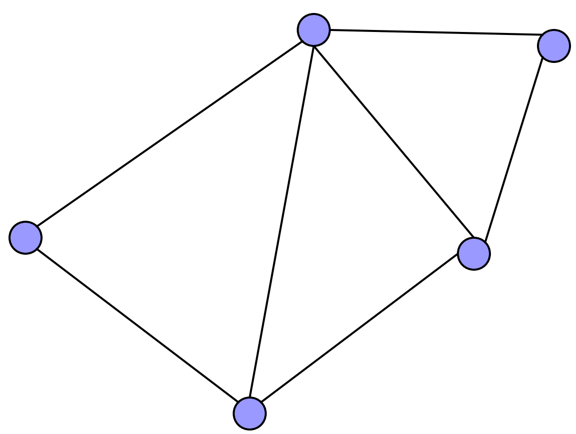
- Total space required: $O(k^2 \varepsilon^{-2} \log^{-1} \delta \log^2 n \log^2 m)$

Outline

- Introduction
- Entropy Upper Bound
- Lower Bounds and Higher Orders
- Random Walks on Graphs

Random Walks on a Graph

- Can define the Graph Random Walk entropy
  - On an undirected unweighted graph, perform a random walk
  - Entropy of the stationary distribution is exactly $H_G = 1/(2|E|) \sum_i d_i \log d_i$, where d_i = degree of node i
  - Suppose we see a stream of edges from this random walk, can we compute $H_G(S)$?
  - With some work, yes!



Additional Wrinkles

Some trouble ahead, we need “distinct sampling”:

- Need to sample uniformly from set of edges, but each edge may occur many times
  - Replace min-wise sampling with min-wise hashing: hash on edge name, sample the one with smallest hash value
- We want to compute d_is, but we may see same edge many times, should only count once to the degree
  - Will replace counting occurrence with approx count distinct
- Multiple occurrences of same edge may bias sampling
  - Reset the counters every time the sampled edge reoccurs

Relative Error

- Important detail: because G is connected, $|E| > n$ (assuming walk visit every node), so $H_G \geq \log(2(1-1/n))$, i.e. at least a constant
- Because of approximations, we end up with an estimator that is $(1 \pm \varepsilon)H_G$, and bounded variance
- Space becomes large: still need $O(\varepsilon^{-2}$ polylog$)$ estimators, each estimator needs space $O(\varepsilon^{-2}$ polylog$)$ for approximate counting.
- Space bound: $O(\varepsilon^{-4} \log^2 n \log^2 \delta^{-1})$ for (ε,δ) approx
  – don't actually need to see a random walk, any ordering and repetition of edges is sufficient

Open Problems

- We have focused on space, speed is important too
  - Current estimator is slow (relative to network line speeds)
  - Maybe use some hashing tricks to speed up – each new token only updates a subset of estimators?
- Generalize the random walk entropy to other settings – weighted graphs?

Conclusions

- Bertinoro is good to visit
- Relatively simple algorithm for (ε,δ) approximating entropy
- Can't improve the ε^{-2} term
- Higher orders are harder for relative error, can do additive error
- Can also do relative error for entropy of random walks

Implementation: Some Details

Maintain (tag1, tok1, rep1), (tag2, tok2, rep2); tag1 < tag2

tok1 will be sample from A, tok2 will be sample from A'

On reading next token, a:

- x = random tag $\in$ [m^3]
- if a == tok1:
  - if x < tag1 then (tag1,tok1,rep1) = (x,a,1) else rep1++
- else:
  - if a == tok2 then rep2++
  - if x < tag1:
    - (tag2,tok2,rep2) = (tag1,tok1,rep1)
    - (tag1,tok1,rep1) = (x,a,1)
  - else:
    - if x < tag2 then (tag2,tok2,rep2) = (x,a,1)

Example Run

Stream:	C	A	A	C
Tags:	0.408	0.815	0.217	0.391

a = [illegible]

x = [illegible]

(tag1,tok1,rep1) = [illegible]

(tag2,tok2,rep2) = [illegible]