Count-Min-Log sketch: Approximately counting with approximate counters

Guillaume Pitel
eXenSa
guillaume.pitel@exensa.com

Geoffroy Fouquier
eXenSa
geoffroy.fouquier@exensa.com

Abstract

Count-Min Sketch [1] is a widely adopted algorithm for approximate event counting in large scale processing. However, the original version of the Count-Min-Sketch (CMS) suffers of some deficiences, especially if one is interested in the low-frequency items, such as in text-mining related tasks. Several variants of CMS [5] have been proposed to compensate for the high relative error for low-frequency events, but the proposed solutions tend to correct the errors instead of preventing them. In this paper, we propose the Count-Min-Log sketch, which uses logarithm-based, approximate counters [7, 4] instead of linear counters to improve the average relative error of CMS at constant memory footprint.

I. Introduction

Count-Min Sketch [1] (CMS) is a widely adopted algorithm for approximate event counting in large scale processing. With proved bounds in terms of mean absolute error and confidence, one can easily design a constant size sketch as an alternative to expensive exact counting for a setting where the total number of event types is approximately known.

CMS is used in many applications, often with a focus on high frequency events [2]. However, in the domain of text-mining, highest frequency events are often of low interest: frequent words are often grammatical, highly polysemous or without any interesting semantics, while low-frequency words are more relevant.

As a matter of fact, a common regularization in text-mining consists in computing the TF-IDF [8] (equation 1) for term/document relevance, and Pointwise Mutual Information [10] (equation 2) or Log-likelihood Ratio [3] between two words in order to estimate the importance of their cooccurrence.

$$idf_i = \log \frac{|D|}{|d_j : t_i \in d_j|} \tag{1a}$$

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i \tag{1b}$$

$$pmi_i, j = \log \frac{p(i,j)}{p(i)p(j)} \tag{2a}$$

In TF-IDF, $|D|$ is the number of documents in the corpus, and $|d_j : t_i \in d_j|$ is the number of documents containing the term t_i. In PMI, $p(i,j)$ is the probability that words i and j appear in a cooccurrence window and $p(i)$ the probability to find i in the corpus.

Those formulas show that higher frequency words will induce a relatively lower value at the end. Moreover, they all use a logarithm, which shows that only the order of magnitude is important.

The original version of the Count-Min-Sketch (CMS) suffers from some deficiences, especially when one is interested not in the high frequency events, but in the low frequency ones, such as in text-mining related tasks. A variant of CMS [5] has been proposed to compensate for the high relative error for low-frequency events, but the solutions explored tend to correct the errors instead of preventing them, i.e., compense over-estimation errors instead of preventing over-estimation itself.

II. Count-Min-Log Sketch with Conservative Update

A sketch is a two-dimensional array $x \times d$. At each depth d corresponds a hash function $h_d \longrightarrow [0, w]$. Each cell of this array is a counter. We propose a variant of the Count-Min Sketch with conservative update that uses logarithm-based, approximate counters [7, 4] instead of linear counters to improve the average relative error of CMS at constant memory footprint. The principle is to use exactly the same structure than Count-Min Sketch, replacing only the classical binary counting cells by log counting cells. With this modification, the Update and Query procedures becomes respectively algorithm 1 and algorithm 2. Note that a similar improvement has been proposed on bloom filter in [9].

The rationale behind this variant lies as follows:

1. The original CMS uses as many bits (usually 32 of 64) to represent low values than high values. However, low values are much more frequent than high values, and they use less bits. As a consequence, one can consider that using smaller counters, and thus increasing the number of counters for the same storage space.

2. Furthermore, and this is especially the case for highly skewed distributions like the ones of Zipfian data, Count-Min Sketch estimates high frequency events very well (Count-Min Sketch with Conservative Update is even better in this regard), but very poorly low-frequency events. As a consequence, estimates of Pointwise Mutual Information, for instance, are largely off

Algorithm 1 Count-Min-Log Sketch UPDATE

Input: sketch width w, sketch depth d , log base $x > 1$, independent hash functions $h_{1..d} : U \to \{1 \dots w\}$, $sk[k, h_k](e)$ return the sketch value of the counter at depth k and a hash produce by the corresponding hash function h_k

```
1: function IncreaseDecision(c)
2:     return True with probability x^{-c}, else False
3: end function
4: function Update(e)
5:     c ← min_{1≤k≤d} sk[k, h_k(e)]
6:     if IncreaseDecision(c) then
7:         for k ← 1...d do
8:             if sk[k, h_k(e)] = c then
9:                 sk[k, h_k(e)] ← c + 1
10:            end if
11:        end for
12:    end if
13: end function
```

for low frequency items, which may cause several problems for large scale NLP tasks.

Algorithm 2 Count-Min-Log Sketch QUERY

Input: sketch width w, sketch depth d , log base $x > 1$, independent hash functions $h_{1..d} : U \to \{1 \dots w\}$

```
1: function PointValue(c)
2:     if c = 0 then
3:         return 0
4:     else
5:         return x^{c-1}
6:     end if
7: end function
8: function Value(c)
9:     if c ≤ 1 then
10:        return PointValue(c)
11:    else
12:        v ←PointValue(c + 1)
13:        return (1−v)/(1−x)
14:    end if
15: end function
16: function Query(e)
17:    c ← min_{1≤k≤d} sk[k, h_k(e)]
18:    return Value(c)
19: end function
```

These observations led to a first conclusion, shared by [5], which is that for NLP tasks, the error metric that should be used on Point Query estimations is the Average Relative Error (ARE), not the Root Mean Square Error (RMSE). This led in turn to the following hypothesis: replacing the linear counters by logarithmic counters should incur, under some conditions, an improvement over the ARE.

Another way to evaluate the precision of an approximate counting sketch for NLP task consists in directly measuring the error on the Pointwise Mutual Information.

III. Empirical evaluation

I. Data and variants

We have verified this hypothesis empirically in the following setting: we count unigrams and bigrams of 500000 words of the 20newsgroups corpus [6]. The small corpus analyzed contains 233k counted elements, composed of 183k bigrams and 50k unigrams. We compare the estimates of three sketches :

CMS-CU is the classical linear Count-min Sketch with Conservative Update,

CMLS16-CU is the Count-min-log Sketch with Conservative Update using a logarithmic base of 1.00025 and 16bits counters, and

CMLS8-CU is the Count-min-log Sketch with Conservative Update using a logarithmic base of 1.08 and 8bits counters.

II. Error on counts

The results for the Average Relative Error on simple counts are shown on figure 1. The vertical line indicates the storage needed to memorize perfectly all the counts (the extra memory required for accessing the counters is not taken into account).

This experiments show that before the perfect storage mark, the estimate error of the CMLS16-CU is approximately 2 to 4 times lower than the error of the estimate of the CMS-CU. The CMLS8-CU error improvement over CMS-CU is in the range of 7 to 12 times, however, the CMLS8-CU reaches a minimal ARE of $10^{-1.5}$ and stops improving, due to the residual error caused by approximate counting.

III. Error on PMI

In a second step, we compute the Pointwise Mutual Information of the bigrams, and the error between the estimated PMI using counts from the sketch versus the exact count. The results for RMSE on estimated PMI are illustrated in figure 2.

These results show that, with sketches near or smaller than the theoretical size of a perfect storage, CMLS16-CU (respectively CMLS8-CU) outperforms CMS-CU by a factor of about 4 (resp. 10) on the RMSE of the PMI.

The histograms of PMI values for each sketch are illustrated by figure 3, showing that for equivalent storage, the CMS-CU presents a very distorted histogram on the right part (the interesting part for NLP tasks), while the CMLS8-CU is much closer to the reference.

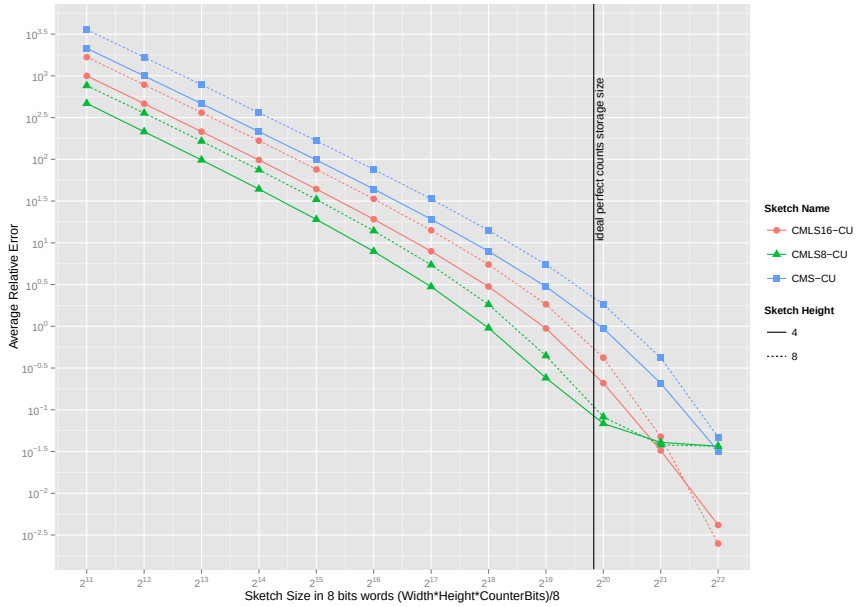

Figure 1: *Average Relative Error of estimated counts with Count-Min Sketch, Count-Min-Log 16bits and Count-Min-Log 8bits.*

IV. Conclusion and perspectives

We have proposed a simple variant of the classical Count-Min-Sketch with Conservative Update which present significant improvement for counts Average Relative Error and RMSE on the Pointwise Mutual Information. While the gain is always clear for high pressure setups, when the available storage is less or equal the ideal storage size, the residual error due to the approximate counting is an absolute lower bound of the error, which can be hit at different sketch sizes depending on the logarithm base.

The next steps of this work are the following:

- Compare results of PMI estimations only for interesting values (i.e. over a given threshold), since we have observed in figure 3 that CMS-CU seems to be particularly far from the reference on the right side of the histogram.
- Evaluate the speed difference of our variant compared to CMS-CU.

Additionally, we are investigating two other directions:

1. Hierarchical storage cells with more cells to store least significant bits and less cells for most significants bits.
2. Probabilistic update rule: we have observed that the ratio between smallest and second smallest estimates is correlated with the error. We want to try a probabilistic update rule that will take this ratio into account.

References

[1] Graham Cormode and S Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[2] Graham Cormode and S Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems (TODS)*, 30(1):249–278, 2005.

[3] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74, 1993.

[4] Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985.

[5] Amit Goyal, Hal Daumé III, and Graham Cormode. Sketch algorithms for estimating point queries in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1093–1103. Association for Computational Linguistics, 2012.

[6] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[7] Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.

[8] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1983.

[9] David Talbot. Succinct approximate counting of skewed data. In *IJCAI*, pages 1243–1248. Citeseer, 2009.

[10] Yan Xu, Gareth JF Jones, JinTao Li, Bin Wang, and Chun-Ming Sun. A study on mutual information-based feature selection for text categorization. *Journal of Computational Information Systems*, 3(3):1007–1012, 2007.

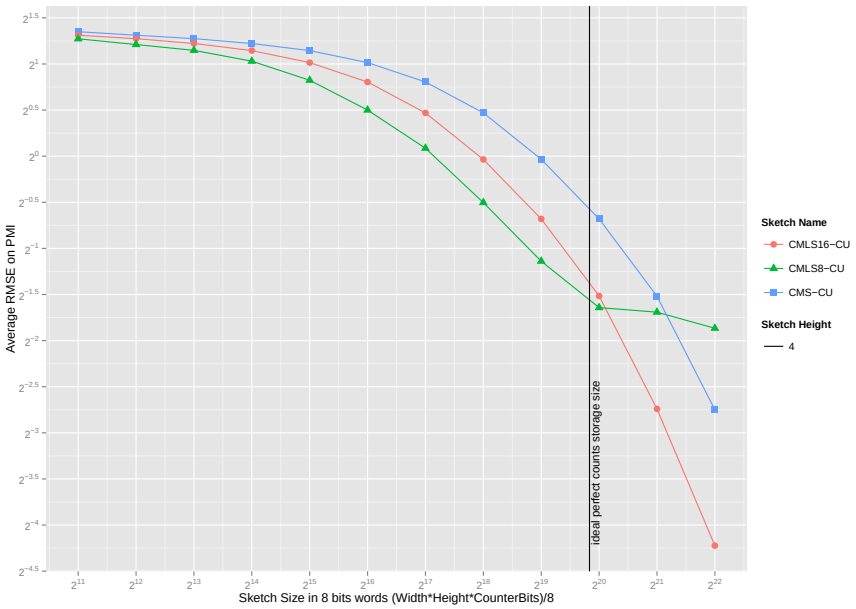

Figure 2: *Root Mean Square Error of estimated PMI with Count-Min Sketch, Count-Min-Log 16bits and Count-Min-Log 8bits.*

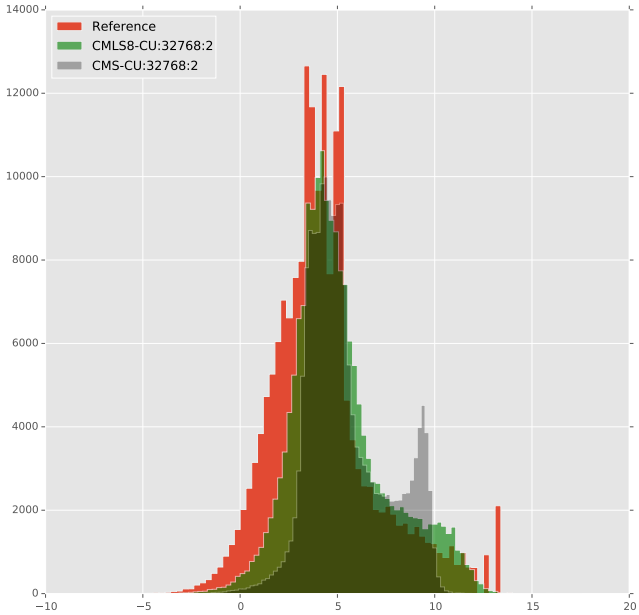

Figure 3: *Histograms of PMI values estimated for CMS-CU and CMLS8-CU sketches with 32kb storage, 2 levels*