Engineering a Simplified 0-Bit Consistent Weighted Sampling

Edward Raff[1,2] Jared Sylvester[1] Charles Nicholas[2]

[1]Booz Allen Hamilton
[2]Univ. of Maryland, Baltimore County

October 25, 2018

Problem: Searching for Weighted Sets

- For a given document d in some set of documents $\mathcal{D}$, we want to be able to find similar documents quickly.
- Total number of possible features F may be large, so we need to minimize both compute and storage costs.
- Each feature $z \in d$ will have some weight associated with it. If $w(d, z) \geq 0$ is the weight of the feature, we'll use the Weighted Jaccard Similarity as our metrics (1).
- How do we make this as fast and space efficient as possible?

$$WJS(S, O) = \frac{\sum_{\forall z \in S \cup O} \min(w(S, z), w(O, z))}{\sum_{\forall z \in S \cup O} \max(w(S, z), w(O, z)))} \tag{1}$$

Min-Hash Approach!

Let's use Min-Hashing to create compact representations of our documents! We choose a hash size K to trade off between accuracy and speed/storage.

Algorithm 1 MinHash Approximation

Require: Two sets S and O that we want to compute the similarity of.

$s \leftarrow 0$
for $k \in 1, 2, \ldots, K$ **do**
    if $\text{Minhash}(S, k) = \text{Minhash}(O, k)$ **then**
        $s \leftarrow s + 1$
    end if
end for
return s/K

ICWS Algorithm

- The Improved Confidence Weighted Sampling (ICWS) [1] is the seminal algorithm for this problem.
- Computing the ICWS Min-Hash is expensive.
    - Requires $\mathcal{O}(KD)$ steps per datapoint
    - Each step requires five logarithms, two exponentiations, and four multiplications/divisions
- Benefit is provably accurate results, but results are not useful if we can't apply the algorithm.
- We want to work towards applications that have *millions* of features per document [3].

Introducing Simplified Consistent Weighted Sampling

- Li [2] showed that the second half of the ICWS hash could be thrown away with minimal impact on accuracy. This reduces memory usage by half, but does the same amount of work.
- Our idea: carry Li's work further. If we are throwing away the second part of the hash, let's exploit that to simplify the whole procedure.

ICWS with Zero Bits

Algorithm 2 Zero-Bit ICWS [1]

procedure MINHASH(Weighted Set S, hash index k)
  for all $z \in S$ **do**
    Seed PRNG with tuple (z, k)
    $r_z \sim \text{Gamma}(2, 1)$
    $c_z \sim \text{Gamma}(2, 1)$
    $\beta_z \sim \text{Uniform}(0, 1)$
    $t_z \leftarrow \left\lfloor \frac{\log w(S,z)}{r_z} + \beta_z \right\rfloor$
    $y_z \leftarrow \exp(r_z(t_z - \beta_z))$
    $a_z \leftarrow \frac{c_z}{y_z \exp(r_z)}$
  end for
  $z^* \leftarrow \arg\min_z a_z$
  $y^* \leftarrow y_{z^*}$
  return z^* ▷ We dropped t_{z^*} following [2]
end procedure

Simplified Consistent Weighted Sampling

Intuition

Normal ICWS uses the tuple (z^*, t_{z^*}) as its hash, and the Zero-Bit version just drops the t_{z^*} portion. If we can obtain similar results without t_{z^*}, why compute it at all?

If we remove the floor operation $\lfloor \cdot \rfloor$, we can simplify the math dramatically. We lose the proof and we may change the results, but does it matter?

$$\begin{aligned}
y_z &= \exp\left(r_z\left(\frac{\log w(S,z)}{r_z} + \beta_z - \beta_z\right)\right) \\
&= \exp\left(r_z\left(\frac{\log w(S,z)}{r_z}\right)\right) \\
&= \exp\left(\log w(S,z)\right) = w(S,z)
\end{aligned}$$

Partial Simplification

Down to just one exponentiation and two multiplications/divisions. However, four samples from the uniform distribution and an additional four logarithms are still needed to produce c_z and r_z.

---

procedure MINHASH(Weighted Set S, hash index k)
    for all $z \in S$ **do**
        Seed PRNG with tuple (z, k)
        $r_z \sim \text{Gamma}(2, 1)$
        $c_z \sim \text{Gamma}(2, 1)$
        $a_z \leftarrow \frac{c_z}{\exp(r_z)} w(S, z)^{-1}$
    end for
    $z^* \leftarrow \arg\min_z a_z$
    $y^* \leftarrow y_{z^*}$
    return z^*
end procedure

Simplification, Step 2

Brainwave!

The term $\frac{c_z}{\exp(r_z)}$ is now *independent* of the value of $w(S, z)^{-1}$! If we can sample from the random distribution defined by this term directly, we can make life even easier.

- Define a pool T of values pre-sampled from the distribution $\frac{c_z}{\exp(r_z)}$
- Pool can be fixed at compile-time and re-used for any input.
- Select a value from the pool based on the tuple (z, k)
- Gets us down to just *one FLOP per hash.*

SCWS Algorithm

Algorithm 3 Simplified CWS (SCWS)

Require: An array T of length $|T|$, where $T[i] \sim c_z \exp(-r_z)$, and large primes p_1 and p_2

```
procedure MINHASH(Weighted Set S, hash index k)
    b ← k · p_2
    for all z ∈ S do
        γ ← (z · p_1 + b) mod |T|          ▷ LCG style index selection
        a_z ← w(S, z)^(-1) · T[γ]          ▷ The only FLOP needed
    end for
    z* ← arg min_z a_z
    return z*
end procedure
```

Evaluating SCWS

We have developed a new algorithm for the weighted min-hash problem. How well does it work? We test this empirically since we lack a bound on its ability.

- How well does this work for word-similarity benchmark?
- As a feature set for learning classification problems?
- Precision at returning nearest neighbors in a search?

Word Similarity, with respect to sketch size K

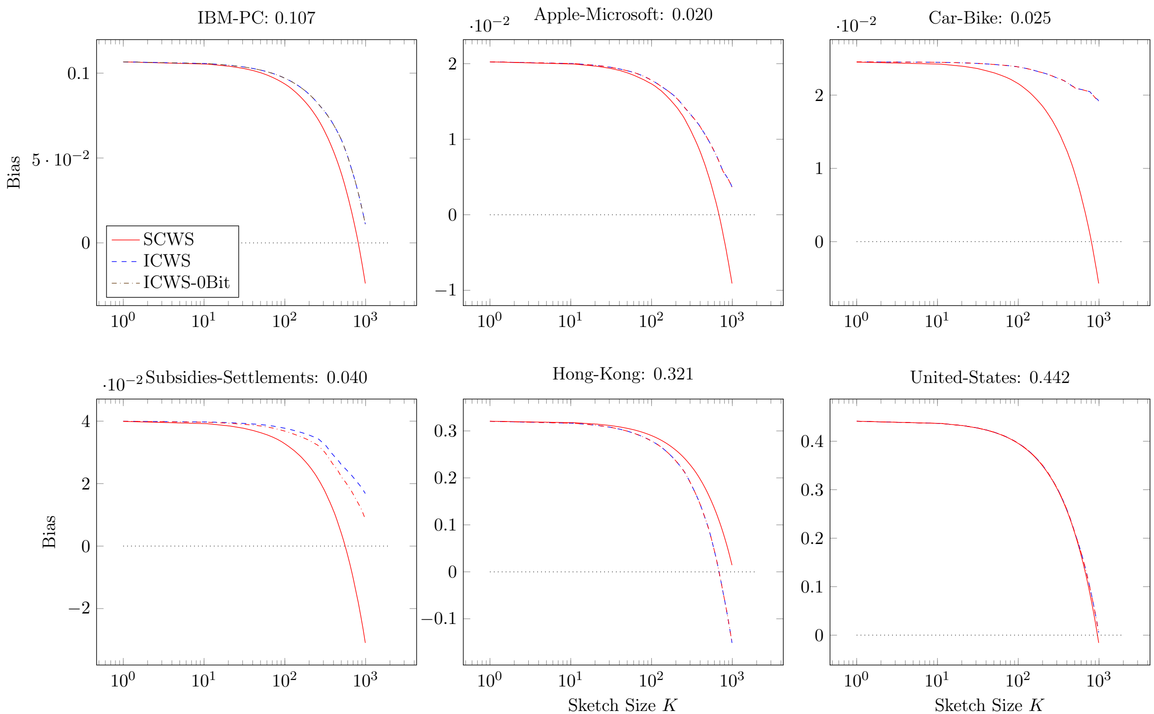


Figure 1: Difference between each CWS algorithm, and the true WJS. The dotted black line shows the value of zero for a perfect estimate and our new SCWS is in red. Above each figure is the word-pair under test, with the true WJS.

Word Similarity, with respect to sketch time (ms)

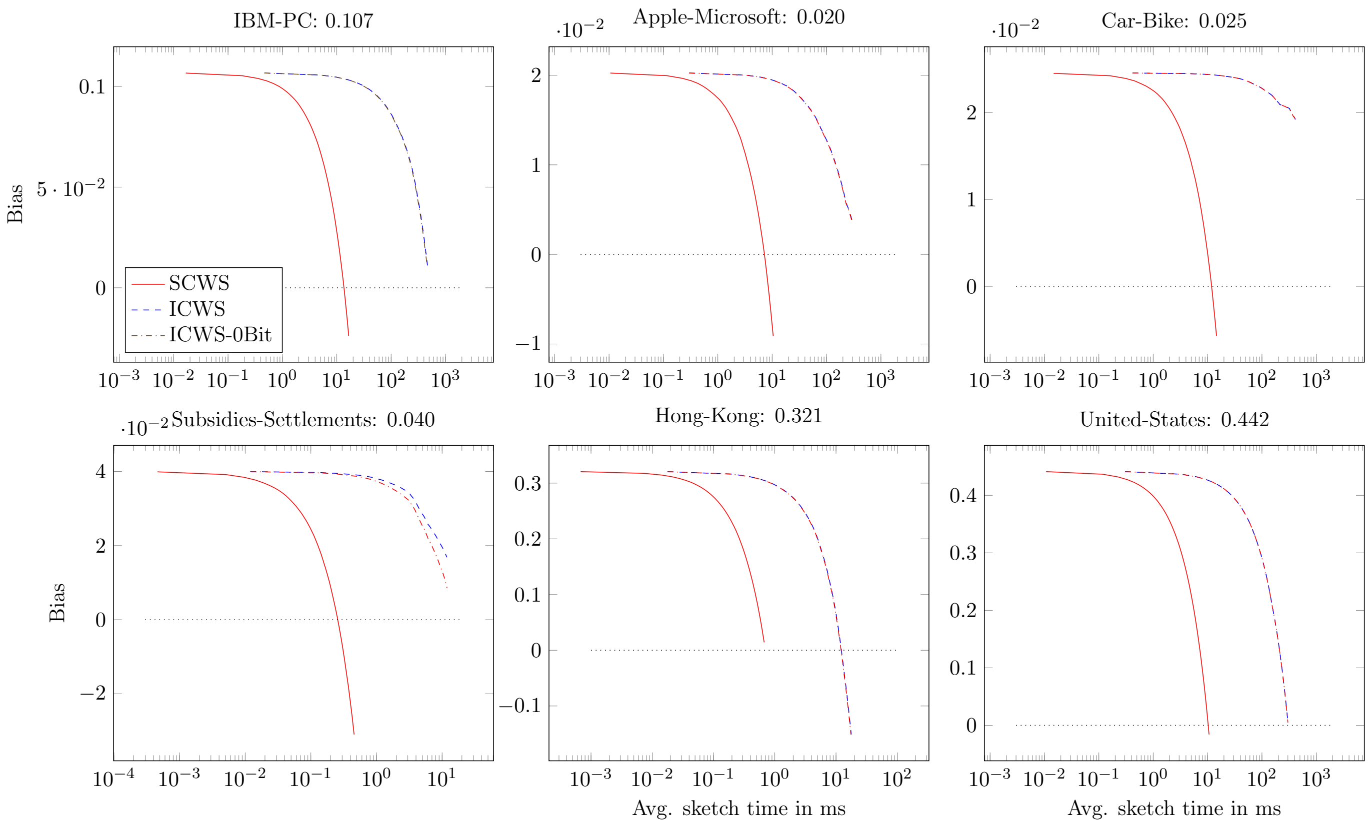


Figure 2: Same as 1, but x-axis replaced with average time to construct the sketch (in milliseconds).

Classification, Accuracy

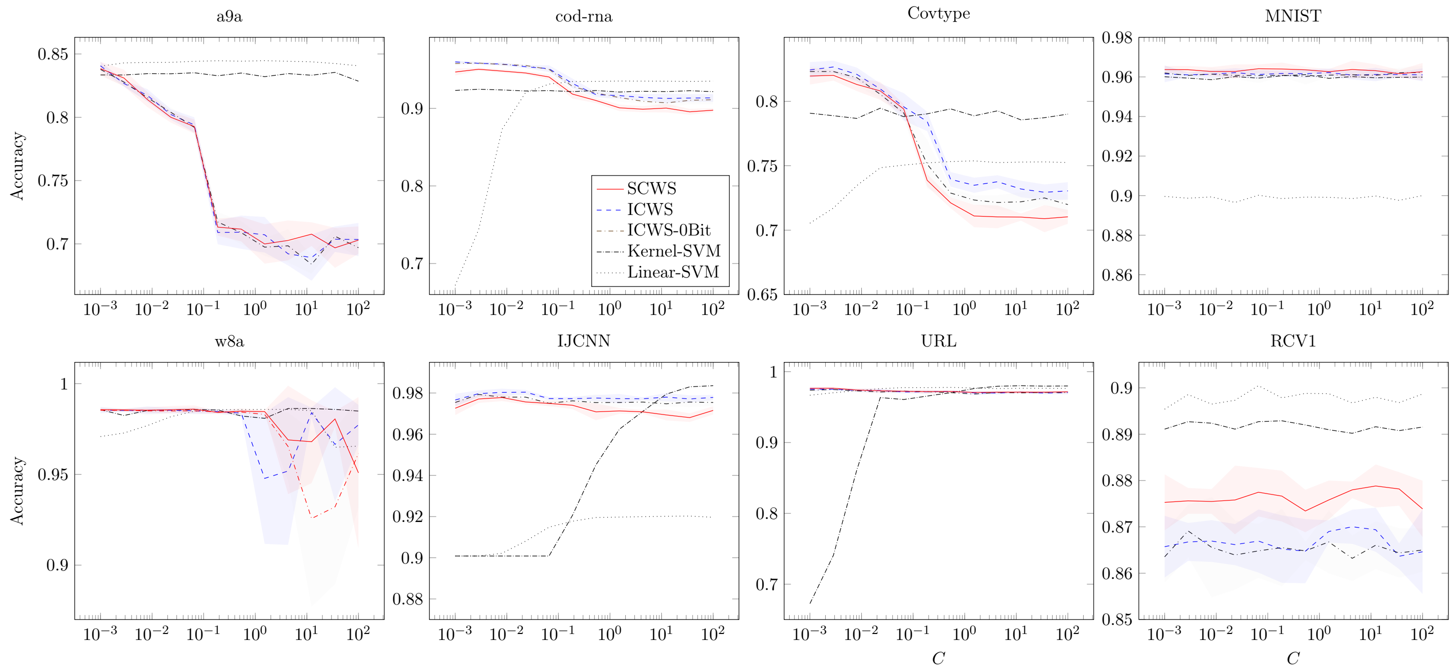


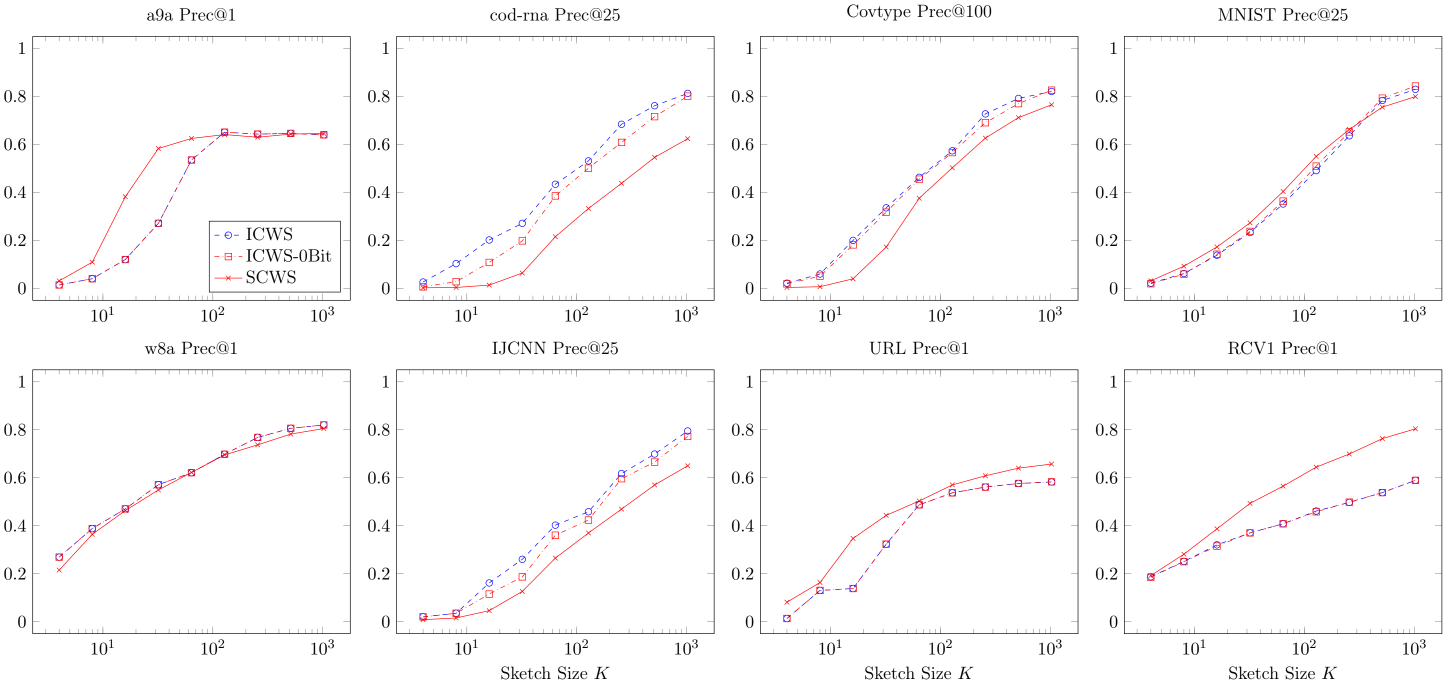
Figure 3: Performance of linear models built from CWS algorithms, compared to a linear and kernel SVM. The x-axis C shows the regularization parameter's value, and the y-axis shows the accuracy of 5-fold cross validation.

Speedup

Table 1: D Indicates the dimension of the dataset, and 'Density' the percentage of non-zero values in the corpus. The right-most column shows how many times faster SCWS was.

			Time to Hash (seconds)			
Dataset	D	Density	ICWS	ICWS-0Bit	SCWS	Speedup
a9a	123	11.3	186	185	9	19.9
cod-rna	8	99.8	106	107	11	9.0
covtype	54	22.1	160	160	12	13.1
MNIST	780	19.2	1,937	1,922	86	22.1
ijcnn1	22	59.1	173	175	22	7.7
w8a	300	3.9	161	162	10	15.4
RCV1	47,236	0.14	661	658	52	12.6
URL	3,231,961	0.004	1,516	1,491	105	14.6

Search Precision



Conclusions

- SCWS performs about as well as ICWS. Sometimes better, sometimes worse, sometimes the same.
- SCWS is usually at least 10x faster, up to 22x.
- SCWS lacks the same theory and backing; might not be appropriate for all cases.

Questions?

raff_edward@bah.com
@EdwardRaffML

sylvester_jared@bah
@jsylvest

nicholas@umbc.edu
What's Twitter?

References I

- Sergey Ioffe. "Improved Consistent Sampling, Weighted Minhash and L1 Sketching". In: *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. Washington, DC, USA: IEEE Computer Society, Dec. 2010, pp. 246–255. ISBN: 978-0-7695-4256-0. DOI: 10.1109/ICDM.2010.80. URL: http://ieeexplore.ieee.org/document/5693978/%20http://dx.doi.org/10.1109/ICDM.2010.80.
- Ping Li. "0-Bit Consistent Weighted Sampling". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. New York, NY, USA: ACM, 2015, pp. 665–674. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2783406. URL: http://doi.acm.org/10.1145/2783258.2783406.

References II

Edward Raff and Charles Nicholas. "Malware Classification and Class Imbalance via Stochastic Hashed LZJD". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security.* AISec '17. New York, NY, USA: ACM, 2017, pp. 111–120. ISBN: 978-1-4503-5202-4. DOI: 10.1145/3128572.3140446. URL: http://doi.acm.org/10.1145/3128572.3140446.