Engineering a Simplified 0-Bit Consistent Weighted Sampling

Edward Raff¹,²  Jared Sylvester¹  Charles Nicholas²

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

October 25, 2018

Booz | Allen | Hamilton
Problem: Searching for Weighted Sets

- For a given document $d$ in some set of documents $D$, 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?

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

1: $s \leftarrow 0$
2: for $k \in 1, 2, \ldots, K$ do
3: if Minhash($S, k$) = Minhash($O, k$) then
4: $s \leftarrow s + 1$
5: end if
6: end for
7: return $s/K$
The Improved Confidence Weighted Sampling (ICWS) [1] is the seminal algorithm for this problem. Computing the ICWS Min-Hash is expensive.

- Requires $O(KD)$ 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].
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.
Algorithm 2 Zero-Bit ICWS [1]

1: procedure Minhash(Weighted Set $S$, hash index $k$)
2:     for all $z \in S$ do
3:         Seed PRNG with tuple $(z, k)$
4:         $r_z \sim \text{Gamma}(2, 1)$
5:         $c_z \sim \text{Gamma}(2, 1)$
6:         $\beta_z \sim \text{Uniform}(0, 1)$
7:         $t_z \leftarrow \left\lfloor \frac{\log w(S, z)}{r_z} + \beta_z \right\rfloor$
8:         $y_z \leftarrow \exp(r_z(t_z - \beta_z))$
9:         $a_z \leftarrow \frac{c_z}{y_z \exp(r_z)}$
10:     end for
11:     $z^* \leftarrow \arg \min_z a_z$
12:     $y^* \leftarrow y_{z^*}$
13:     return $z^*$ \hspace{1cm} \triangleright \text{We dropped } t_{z^*} \text{ following [2]}
14: 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 \([\cdot]\), we can simplify the math dramatically. We lose the proof and we may change the results, but does it matter?

\[
y_z = \exp \left( r_z \left( \frac{\log w(S, z)}{r_z} + \beta_z - \beta_z \right) \right)
\]

\[
= \exp \left( r_z \left( \frac{\log w(S, z)}{r_z} \right) \right)
\]

\[
= \exp \left( \log w(S, z) \right) = w(S, z)
\]
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 $r_z$.

---

1: **procedure** MINHASH(Weighted Set $S$, hash index $k$)
2:   
3:     for all $z \in S$ do
4:        Seed PRNG with tuple $(z, k)$
5:         $r_z \sim \text{Gamma}(2, 1)$
6:         $c_z \sim \text{Gamma}(2, 1)$
7:         $a_z \leftarrow \frac{c_z}{\exp(r_z)} w(S, z)^{-1}$
8:     end for
9: $z^* \leftarrow \text{arg min}_z a_z$
10: $y^* \leftarrow y_{z^*}$
11: **return** $z^*$
12: **end procedure**
Brainwave!
The term $\frac{c_z \exp(r_z)}{\exp(r_z)}$ 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.
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 $p_2$

1: procedure Minhash(Weighted Set $S$, hash index $k$)
2: $b \leftarrow k \cdot p_2$
3: for all $z \in S$ do
4: $\gamma \leftarrow (z \cdot p_1 + b) \mod |T|$ ▶ LCG style index selection
5: $a_z \leftarrow w(S, z)^{-1} \cdot T[\gamma]$ ▶ The only FLOP needed
6: end for
7: $z^* \leftarrow \arg \min_z a_z$
8: return $z^*$
9: 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.
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.
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.

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