Sketching Techniques for Large Scale NLP

Amit Goyal, Jagadeesh Jagarlamudi,
Hal Daumé III, and Suresh Venkatasubramanian
email: \{amitg,jags,hal,suresh\}@cs.utah.edu
web: http://www.cs.utah.edu/~amitg/

NAACL-HLT 2010
6th Web as Corpus Workshop
5th June 2010
Large amounts of Data

Problem

- Huge quantity of text available
- New text coming every day like news, messages, e-mails, and twitter
- Requires counting items
- Storing all counts is important as low frequency counts matters
- Need lots of disk space and memory

Solution

Approximate randomized Algorithms

Trades off disk space and memory usage with accuracy of counts
Large amounts of Data

Problem
- Huge quantity of text available
- New text coming every day like news, messages, e-mails, and twitter
- Requires counting items
- Storing all counts is important as low frequency counts matters
- Need lots of disk space and memory

Solution
- Approximate randomized Algorithms
- Trades off disk space and memory usage with accuracy of counts
Conventional counts storing

1980 ... 2010
Text

Storage
Small-spaced counts storing Approach

1980 ... 2010
Text

Small Model
Batch counts storing Approach

1980...2010 Text + 2010...2020 Text

Merge

1980...2020 Text

Small Model
Online counts storing Approach

What properties we want from the ideal model?

- Small-Space Model
- Updated Online
- Constant Update Time
- Constant Query Time
- Easily Parallelizable (e.g. using MapReduce Framework)
- Easy to merge counts
Approximate Algorithms in NLP

- Bloom filters and perfect hashing: Compresses $n$-gram storage
  [Talbot and Osborne; ACL 2007] [Talbot and Brants; ACL 2008]

- Deterministic Streaming Algorithm: Finding Top $K$ $n$-grams
  [Goyal et al.; NAACL 2009]

- Bloom Filters with Morris Counting: Online storing counts
  [Talbot; IJCAI 2009] [Van Durme and Lall; IJCAI 2009]

- Building Top-$K$ ranklists based on PMI
  [Van Durme and Lall; NIPS 2009]
Bloom filters and perfect hashing: Compresses $n$-gram storage
[Talbot and Osborne; ACL 2007] [Talbot and Brants; ACL 2008]

Deterministic Streaming Algorithm: Finding Top $K$ $n$-grams
[Goyal et al.; NAACL 2009]

Bloom Filters with Morris Counting: Online storing counts
[Talbot; IJCAI 2009] [Van Durme and Lall; IJCAI 2009]

Building Top-$K$ ranklists based on PMI
[Van Durme and Lall; NIPS 2009]

Online Generation of Locality Sensitive Hash Signatures
[Van Durme and Lall; To appear in ACL 2010]

Sketch techniques for scaling Distributional Similarity to the Web
[Goyal et al.; To appear in ACL 2010 workshop GEMS ]
Storing all word pairs from Large corpora

Storing all word pairs from Large Corpora is memory intensive

Increase in data $\implies$ More memory usage

![Graph showing log2 of number of unique items vs log2 of number of words]
Storing all word pairs from Large corpora

Storing all word pairs from Large Corpora is memory intensive

Increase in data $\implies$ More memory usage

**GOAL:** Store all word pairs counts in small-space
Given: Input stream of $N$ word pairs i.e. $wp_1 \ldots wp_N$

- **Sketch**: Summary data structure to store streaming word pairs in small-space
- **Hashing**: to map word pairs onto the small-space sketch vector
- **Streaming algorithm**: read from left to right without going backwards
- **Requires space**: smaller than the size of the input stream
Count-Min Sketch (CM) \cite{CormodeMuthukrishnan:2002}

\[
\begin{bmatrix}
\text{sketch}[1,1] & \cdots & \text{sketch}[1,w] \\
\vdots & \ddots & \vdots \\
\text{sketch}[d,1] & \cdots & \text{sketch}[d,w]
\end{bmatrix}
\]

- Size of matrix $= w \times d$
- $w = \frac{2}{\epsilon}$ and $d = \log\left(\frac{1}{\delta}\right)$
- $(\epsilon, \delta)$ controls amount of error and probability of failure
- $d$ hash functions $h_1, \ldots, h_d$: chosen uniformly random from pairwise independent family
- Each hash maps $\{wp_1 \ldots wp_N\}$ onto $1 \ldots w$
Update CM Sketch: \( wp_1 \) arrives

Size of Matrix= \( w \times d \); Set \( w = 7 \) and \( d = 3 \)
Update CM Sketch: $wp_2$ arrives

$$
\begin{align*}
\text{Update CM Sketch: } & wp_2 \text{ arrives} \\
\end{align*}
$$
Query CM Sketch for $wp_2$

$h_1(wp_2) = 1$

$h_2(wp_2) = 4$

$h_3(wp_2) = 6$

$Query(wp_2) = \min(2, 1, 2) = 1$
Query CM Sketch for $wp_1$

$\begin{array}{c}
wp_1 \\
h_1(wp_1) = 1 \\
h_2(wp_1) = 3 \\
h_3(wp_1) = 6
\end{array}$

$\begin{array}{cccccccc}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 0
\end{array}$

$Query(wp_1) = min(2, 1, 2) = 1$
Combine with CM Sketch to decrease over estimation error

- Suppose $wp_i$ arrives
- Current estimate of $wp_i = \hat{c} = \text{Query}(wp_i)$
- Counts are updated according to: $\forall 1 \leq k \leq d$

$$\text{sketch}[k, h_k(w)] \leftarrow \max\{\text{sketch}[k, h_k(w)], \hat{c} + 1\}$$

- Do not increase counts if they are already big “enough”
Update CM Sketch: $wp_i$ arrives

\[
\begin{array}{ccccccc}
3 & 3 & 4 & 5 & 7 & 2 & 1 \\
2 & 3 & 8 & 4 & 3 & 1 & 4 \\
6 & 4 & 2 & 3 & 1 & 7 & 2 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
3 & 3 & 4 & 6 & 7 & 2 & 1 \\
2 & 3 & 9 & 4 & 3 & 1 & 4 \\
7 & 4 & 2 & 3 & 1 & 7 & 2 \\
\end{array}
\]

$w$

$\begin{array}{ccccccc}
3 & 3 & 4 & 5 & 7 & 2 & 1 \\
2 & 3 & 8 & 4 & 3 & 1 & 4 \\
6 & 4 & 2 & 3 & 1 & 7 & 2 \\
\end{array}$

$\begin{array}{ccccccc}
3 & 3 & 4 & 6 & 7 & 2 & 1 \\
2 & 3 & 9 & 4 & 3 & 1 & 4 \\
7 & 4 & 2 & 3 & 1 & 7 & 2 \\
\end{array}$

$w$ + $wp_i$

$h_1(wp_i) = 4$

$h_2(wp_i) = 3$

$h_3(wp_i) = 1$
Query CM Sketch for $wp_j$

$$h_1(wp_j) = 5$$

$$h_2(wp_j) = 3$$

$$h_3(wp_j) = 1$$

$$Query(wp_j) = \min(7, 9, 7) = 7$$
Conservative update CM (CU) Sketch: \( wp_i \) arrives

\[
\begin{array}{cccccc}
3 & 3 & 4 & 5 & 7 & 2 \\
2 & 3 & 8 & 4 & 3 & 1 \\
6 & 4 & 2 & 3 & 1 & 7 \\
\end{array}
\]

\[
\begin{array}{cccccc}
3 & 3 & 4 & 6 & 7 & 2 \\
2 & 3 & 8 & 4 & 3 & 1 \\
6 & 4 & 2 & 3 & 1 & 7 \\
\end{array}
\]

\( w \)

\( d \)

\( h_1(wp_i) = 4 \)

\( h_2(wp_i) = 3 \)

\( h_3(wp_i) = 1 \)
Query CU Sketch for $wp_j$

$$h_1(wp_j) = 5$$

$$h_2(wp_j) = 3$$

$$h_3(wp_j) = 1$$

$$Query(wp_j) = \min(7, 8, 6) = 6$$
Linearity property of Sketch enables parallelization

Use same set of hash functions

\[
\begin{array}{cccc}
1 & 2 & 3 & 1 \\
2 & 1 & 0 & 3 \\
5 & 1 & 1 & 2 \\
\end{array}
\quad +
\quad \begin{array}{cccc}
1 & 2 & 3 & 1 \\
3 & 2 & 0 & 4 \\
6 & 2 & 3 & 3 \\
\end{array}
\quad =
\quad \begin{array}{cccc}
2 & 4 & 6 & 2 \\
5 & 3 & 0 & 7 \\
11 & 3 & 4 & 5 \\
\end{array}
\]

A \quad B \quad A+B
Algorithm Guarantees and Properties

Algorithm Guarantees:

- Time for updates $O(d)$
- Query Time $O(d)$
- Space used $O(w \times d)$
- Reported counts are at most $\epsilon N$ greater than Actual counts with failure probability $\delta$
Algorithm Guarantees and Properties

Algorithm Guarantees:

- Time for updates $O(d)$
- Query Time $O(d)$
- Space used $O(w \times d)$
- Reported counts are at most $\epsilon N$ greater than Actual counts with failure probability $\delta$

Algorithm Properties:

- Approximate counts by not explicitly storing the word pairs
- Sublinear space data structure
- Constant Query and Update Time
- Conservative Update (CU) reduces over estimation in counts
- Easily Parallelizable using MapReduce Framework
Evaluate the quality of CM and CU Sketch counts

Data-set: 2 million sentences from Gigaword

Compute counts of word pairs over sliding window of size 14

Stream size of words/word pairs is 230 million

Store counts of all words and word pairs in Sketch
Compare CM and CU counts with Exact counts

Tradeoff between \( w = \frac{50M}{d} \) and \( d \) with fixed array size of 50M counters

Most of the errors occur on low frequency items.
Error is close to zero for frequent items.
CU reduces relative error by more than a factor of 2.
Compare CM and CU counts with Exact counts

Tradeoff between $w = \frac{50M}{d}$ and $d$ with fixed array size of 50M counters

- Most of the errors occur on low frequency items
- Error is close to zero for frequent items
- CU reduces relative error by more than a factor of 2
- Having 3 hash functions gives lowest error
Evaluating word association rankings of word pairs

- Pointwise Mutual Information
- Convert **CU counts** into approximate PMI between word pairs
- Use 3 hash functions
- Compare CU PMI word pairs ranking with Exact PMI ranking
- Evaluation Metric: Accuracy and Spearmans correlation ($\rho$)
- Data-set: 2 million sentences from Gigaword
Evaluating PMI rankings of word pairs

Compare word pairs association rankings obtained using approximate PMI with sketch and exact counts

<table>
<thead>
<tr>
<th>Counters</th>
<th>50M</th>
<th>100M</th>
<th>200M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top K</td>
<td>Acc</td>
<td>ρ</td>
<td>Acc</td>
</tr>
<tr>
<td>100</td>
<td>.18</td>
<td>.31</td>
<td>.77</td>
</tr>
<tr>
<td>200</td>
<td>.21</td>
<td>.68</td>
<td>.73</td>
</tr>
<tr>
<td>500</td>
<td>.24</td>
<td>.31</td>
<td>.71</td>
</tr>
<tr>
<td>1000</td>
<td>.33</td>
<td>.17</td>
<td>.74</td>
</tr>
</tbody>
</table>

Higher is better
Evaluating PMI rankings of word pairs

Compare word pairs association rankings obtained using approximate PMI with sketch and exact counts

<table>
<thead>
<tr>
<th>Counters</th>
<th>50M</th>
<th>100M</th>
<th>200M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>ρ</td>
<td>Acc</td>
</tr>
<tr>
<td>Top K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>.18</td>
<td>.31</td>
<td>.77</td>
</tr>
<tr>
<td>200</td>
<td>.21</td>
<td>.68</td>
<td>.73</td>
</tr>
<tr>
<td>500</td>
<td>.24</td>
<td>.31</td>
<td>.71</td>
</tr>
<tr>
<td>1000</td>
<td>.33</td>
<td>.17</td>
<td>.74</td>
</tr>
</tbody>
</table>

Higher is better

200M counter model does well on 230M stream size
Need Counters linear in size of stream
Finding Semantic Orientation (SO) of word

Task Description

- Find whether a word is positive or negative
- SO based on strength of its association with the seven positive words (+), and the seven negative words (-)

\[
SO-PMI(w) = PMI(+, w) - PMI(-, w)
\]

- Same positive and negative words as in [Turney and Littman, 2002]
Finding Semantic Orientation (SO) of word

Task Description

- Find whether a word is positive or negative
- SO based on strength of its association with the seven positive words (+), and the seven negative words (-)

\[
SO-PMI(w) = PMI(+, w) - PMI(-, w)
\]

- Same positive and negative words as in [Turney and Littman, 2002]

Test Set

- General Inquirer lexicon [Stone et al.; 1966]
- Test set: 1619 positive and 1989 negative words
- Accuracy metric: percentage of correctly identified words
- Gigaword corpus [Graff, 2003]
- Copy of web crawled by [Ravichandran et al., 2005]
Training Data

- Gigaword corpus [Graff, 2003]
- Copy of web crawled by [Ravichandran et al., 2005]

- Compute counts of word pairs over sliding window of size 10
- Store counts of all words and word pairs in CU Sketch
- Use 1 and 2 billion counter matrix with 3 hash functions
Training Data

- Gigaword corpus [Graff, 2003]
- Copy of web crawled by [Ravichandran et al., 2005]

- Compute counts of word pairs over sliding window of size 10
- Store counts of all words and word pairs in CU Sketch
- Use 1 and 2 billion counter matrix with 3 hash functions

Counts computed from three data-sets

<table>
<thead>
<tr>
<th>Data-set</th>
<th>Size(GB)</th>
<th>Stream Size(billion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigaWord (Giga)</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>WEB1 (GIGA+50%WEB)</td>
<td>21</td>
<td>38</td>
</tr>
<tr>
<td>WEB2 (GIGA+100%WEB)</td>
<td>37</td>
<td>58</td>
</tr>
</tbody>
</table>
## Results on SO task

<table>
<thead>
<tr>
<th>Model</th>
<th>#of counters</th>
<th>Mem. Usage</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>n/a</td>
<td></td>
<td>Giga 6GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>64.77</td>
</tr>
</tbody>
</table>

Using web data improves the accuracy
Results on SO task

<table>
<thead>
<tr>
<th>Model</th>
<th>#of counters</th>
<th>Mem. Usage</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Giga 6GB</td>
</tr>
<tr>
<td>Exact</td>
<td></td>
<td>n/a</td>
<td>64.77</td>
</tr>
</tbody>
</table>

- Using web data improves the accuracy

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1B 4GB</td>
<td>62.95</td>
<td>73.93</td>
<td>75.03</td>
<td></td>
</tr>
</tbody>
</table>

- Using 1B counters result in only 2% worse compare to Exact
- Less accuracy comes at huge memory saving
Results on SO task

<table>
<thead>
<tr>
<th>Model</th>
<th>#of counters</th>
<th>Mem. Usage</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>n/a</td>
<td></td>
<td>Giga 6GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>64.77</td>
</tr>
</tbody>
</table>

- Using **web data** improves the accuracy

| 1B  | 4GB | 62.95 | 73.93 | 75.03 |

- Using **1B counters** result in only **2% worse** compare to Exact

- Less accuracy comes at **huge memory saving**

| 2B  | 8GB | 64.69 | 75.86 | 76.96 |

- Using **2B counters** result in **similar accuracy** to Exact

- 30 times less counters than stream size does not decrease accuracy
Discussion

Take Home Message

- Sketch to compute approx. word-association measures like PMI
- Efficient, small-footprint method
- Scales to 60 billion stream text using 8 GB main memory
- Constant Query and Update time
- Easily Parallelizable using MapReduce Framework
Discussion

Take Home Message

- Sketch to compute approx. word-association measures like PMI
- Efficient, small-footprint method
- Scales to 60 billion stream text using 8 GB main memory
- Constant Query and Update time
- Easily Parallelizable using MapReduce Framework

Future Work

- Approx. word-association scores for noun clustering, word-sense disambiguation
- Use Sketches for efficient inner product computations
- Work with phrase-pairs
Discussion

Take Home Message

- Sketch to compute approx. word-association measures like PMI
- Efficient, small-footprint method
- Scales to 60 billion stream text using 8 GB main memory
- Constant Query and Update time
- Easily Parallelizable using MapReduce Framework

Future Work

- Approx. word-association scores for noun clustering, word-sense disambiguation
- Use Sketches for efficient inner product computations
- Work with phrase-pairs

Thanks! Questions?