Fast Approximate k-Way Similarity Search

Anshumali Shrivastava
Dept. of Computer Science
Cornell University

Ping Li
Dept. of Statistics & Biostatistics
Dept. of Computer Science
Rutgers University
The 3-way Resemblance

- **Standard Jaccard or 2-way resemblance** is one of the widely used similarity measures over set representations (e.g. $S_1$, $S_2$) of documents defined as:

$$R = Sim(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

- **3-way resemblance** is a natural extension defined over 3 sets as:

$$R^{3\text{way}} = Sim(S_1, S_2, S_3) = \frac{|S_1 \cap S_2 \cap S_3|}{|S_1 \cup S_2 \cup S_3|}$$

- Can also be thought of as **normalized co-occurrence**.
A Simple Experiment with Google Sets

**Problem:** Given two (or a set of words) $w_1$ and $w_2$, complete the set by finding more words representing the set (or words that are *semantically similar*).

**Competing Methods:**

- **Google:** The original *Google’s algorithm* available via Google spreadsheet.
- **3-way resemblance (3-way):** Use 3-way resemblance $\frac{|w_1 \cap w_2 \cap w|}{|w_1 \cup w_2 \cup w|}$ to rank every word $w$ and report top 5 words.
- **Sum Resemblance (SR):** Use the sum of pairwise resemblance $\frac{|w_1 \cap w|}{|w_1 \cup w|} + \frac{|w_2 \cap w|}{|w_2 \cup w|}$ and report top 5 words based on this similarity.
- **Pairwise Intersection (PI):** Retrieve top 100 words based on pairwise resemblance for each $w_1$ and $w_2$ independently. Report the *common* words.

In our experiments, all methods except Google use *binary term-document* representation generated from *1M wikipedia* documents collected from Wikidump.
### Google Sets: Results

<table>
<thead>
<tr>
<th>“JAGUAR” AND “TIGER”</th>
<th>“MILKY” AND “WAY”</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GOOGLE</strong></td>
<td><strong>3-WAY</strong></td>
</tr>
<tr>
<td>LION</td>
<td>LEOPARD</td>
</tr>
<tr>
<td>LEOPARD</td>
<td>CHEETAH</td>
</tr>
<tr>
<td>CHEETAH</td>
<td>LION</td>
</tr>
<tr>
<td>CAT</td>
<td>PANTHER</td>
</tr>
<tr>
<td>DOG</td>
<td>CAT</td>
</tr>
</tbody>
</table>
**Improving Retrieval**

**Problem:** Refine search in presence of more than one representative query.

**Scenarios:**

- **Pairwise:** Just one query $q$, rank element $e$ based on resemblance $\frac{|q \cap e|}{|q \cup e|}$.
- **3-way NNbor:** Two representative queries $q_1$ and $q_2$, rank based on 3-way resemblance $\frac{|q_1 \cap q_2 \cap e|}{|q_1 \cup q_2 \cup e|}$.
- **4-way NNbor:** Three representative queries $q_1, q_2$ and $q_3$, rank based on 4-way resemblance $\frac{|q_1 \cap q_2 \cap q_3 \cap e|}{|q_1 \cup q_2 \cup q_3 \cup e|}$. 
**Improving Retrieval: Results**

Table 1: Percentage of top candidates with the same labels as that of query (queries) retrieved using various similarity criteria. Higher value indicates better retrieval quality.

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>WEBSPAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOP</strong></td>
<td>1  10  20  50</td>
<td>1  10  20  50</td>
</tr>
<tr>
<td><strong>PAIRWISE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>94.20</td>
<td>92.33  91.10  89.06</td>
<td>98.45 96.94 96.46 95.12</td>
</tr>
<tr>
<td>96.90</td>
<td>96.13  95.36  93.78</td>
<td>99.75 98.68 97.80 96.11</td>
</tr>
<tr>
<td>4-WAY</td>
<td>97.70  96.89  96.28  95.10</td>
<td>99.90 98.87 98.15 96.45</td>
</tr>
</tbody>
</table>
### Why 3-way Resemblance?

<table>
<thead>
<tr>
<th></th>
<th>SR</th>
<th>PI</th>
<th>3-way</th>
<th>Custom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality?</td>
<td>Poor</td>
<td>Poor</td>
<td>Looks Good</td>
<td>Say Good</td>
</tr>
<tr>
<td>Efficient?</td>
<td>No</td>
<td>Yes</td>
<td>Yes (this work)</td>
<td>?</td>
</tr>
</tbody>
</table>

Note: Linear run time is *not acceptable* in applications like search.
3-way Search Problems and \(c\)-Approximate Versions

- Given two sets \(S_1\) and \(S_2\), find \(S_3 \in \mathcal{C}\) maximizing \[
\frac{|S_1 \cap S_2 \cap S_3|}{|S_1 \cup S_2 \cup S_3|}.
\] \(O(n)\)

\(c\)-Approximate Version (3-way \(c\)-NN): Given two query sets \(S_1\) and \(S_2\), if there exist \(S_3 \in \mathcal{C}\) with \(Sim(S_1, S_2, S_3) \geq R_0\), then we report some \(S_3' \in \mathcal{C}\) so that \[
\frac{|S_1 \cap S_2 \cap S_3'|}{|S_1 \cup S_2 \cup S_3'|} \geq cR_0
\] with probability \(\geq 1 - \delta\).

- Given set \(S_1\), find sets \(S_2, S_3 \in \mathcal{C}\) maximizing \[
\frac{|S_1 \cap S_2 \cap S_3|}{|S_1 \cup S_2 \cup S_3|}.
\] \(O(n^2)\)

\(c\)-Approximate Version (3-way \(c\)-CP): Given a query set \(S_1\), if there exist a pair of set \(S_2, S_3 \in \mathcal{C}\) with \(Sim(S_1, S_2, S_3) \geq R_0\), then we report sets \(S_2', S_3' \in \mathcal{C}\) so that \[
\frac{|S_1 \cap S_2' \cap S_3'|}{|S_1 \cup S_2' \cup S_3'|} \geq cR_0
\] with probability \(\geq 1 - \delta\).

- Find \(S_1, S_2, S_3 \in \mathcal{C}\) maximizing \[
\frac{|S_1 \cap S_2 \cap S_3|}{|S_1 \cup S_2 \cup S_3|}.
\] \(O(n^3)\)

\(c\)-Approximate Version (3-way \(c\)-BC): If there exist sets \(S_1, S_2, S_3 \in \mathcal{C}\) with \(Sim(S_1, S_2, S_3) \geq R_0\), then we report sets \(S_1', S_2', S_3' \in \mathcal{C}\) so that \[
\frac{|S_1' \cap S_2' \cap S_3'|}{|S_1' \cup S_2' \cup S_3'|} \geq cR_0
\] with probability \(\geq 1 - \delta\).
Key Ideas: Probabilistic Indexing

Given three sets \( S_1, S_2, S_3 \subseteq \Omega \) and an independent random permutation \( \pi : \Omega \rightarrow \Omega \), we have the following:

\[
Pr \left( \min(\pi(S_1)) = \min(\pi(S_2)) = \min(\pi(S_3)) \right) = R^{3way}.
\]

- This estimator leads to an efficient indexing scheme.

- If we map every element \( S \in C \) to the hash bucket indexed by \( B(S) = [\min \pi(S); \min \pi(S)] \), given query \( S_1, S_2 \) we probe only the bucket \( B'(S_1, S_2) = [\min \pi(S_1); \min \pi(S_2)] \) and we do better than random!

- This idea can be converted into a provably fast algorithm for \( c \)-NN search by adding two more handles \( K \) and \( L \) to control the probability.
Main Algorithmic Results

**Theorem 1** For $R^3$way $c$-NN one can construct a data structure with $O(n^\rho \log_{1/cR_0} n)$ query time and $O(n^{1+\rho})$ space.

**Theorem 2** For $R^3$way $c$-CP one can construct a data structure with $O(n^{2\rho \log_{1/cR_0} n})$ query time and $O(n^{1+2\rho})$ space.

**Theorem 3** For $R^3$way $c$-BC there exist an algorithm with running time $O(n^{1+2\rho \log_{1/cR_0} n})$.

Figure 1: Plot of $\rho = 1 - \frac{\log 1/c}{\log 1/c + \log 1/R_0} < 1$ values with respect to $c$ for various thresholds $R_0$. 
Are there more $k$-way similarities which are efficient?

**Theorem 4** Any PGF transformation on 3-way resemblance $R^{3\text{way}}$ admits efficient $c$-NN search.

where $PGF(S) = \sum_{i=1}^{\infty} p_i S^i$ with all $p_i \geq 0$ satisfying $\sum_{i=1}^{\infty} p_i = 1$

**Corollary 1** $e^{R^{3\text{way}}-1}$ admits efficient $c$-NN search.

**Theorem 5** Weighted 3-way resemblance, defined as $Sim(x, y, z) = \sum_i \frac{\min\{x_i, y_i, z_i\}}{\max\{x_i, y_i, z_i\}}$, naturally enjoys all efficiently guarantees of 3-way resemblance using consistent weighted sampling instead of Minhash.
Conclusions

- 3-way (and higher) resemblance seems a natural choice for many interesting search problems, and at the same time it admits efficient search algorithms.

- The idea of probabilistic hashing can reduce the computational requirements significantly.

More Possibilities

Joint Recommendations:

- Users A and B would like to watch a movie together. Profile of each person represented as a binary sparse vector over a giant universe of attributes. For example: actors, actresses, genres, directors, etc, which she/he likes. Represent movie M as binary vectors over the same universe.

- A natural measure to maximize is $\frac{|A \cap B \cap M|}{|A \cup B \cup M|}$. 