Being Prepared In A Sparse World: The Case of KNN Graph Construction

François Taïani – ASAP – Inria Rennes
Co-authors

- Joint work with
  
  Antoine Boutet  
  Nupur Mittal  
  Anne-Marie Kermarrec

- Published at ICDE 2016
Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

Netflix awarded a $1 million prize to a developer team in 2009 for an algorithm that increased the accuracy of the company's recommendation engine by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why: a combination of too much engineering effort for the results, and a shift from movie recommendations to the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.
Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

$1 million prize

recommendation

too much engineering effort
Which **practical approaches for scale and performance?**

- **$1 million prize** for recommendation
- **too much engineering effort**

Netflix awarded a developer team in 2009 for an algorithm that increased the accuracy of the company's movie recommendation engine by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix **announced** on its blog Friday. The post goes on to explain why a combination of too much engineering effort...
The problem: KNN graph construction
The intuition: Is greed all there is?
KIFF: K-nearest neighbor Fast and eFFicient
Evaluation
Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- KIFF: K-nearest neighbor Fast and eFFicient
- Evaluation
KNN Graph Construction

- **Entities** (e.g. users)

- **Similarity function**

- **Goal**: for each entity find $k$ closest entities

- **Many applications**
  - search
  - recommendation,
  - learning, ...

![Diagram showing KNN Graph Construction with entities, similarity function, and applications](image-url)
Challenges

- Brute force not scalable: $O(n^2 \times \log(k))$

- Alternatives: Approximate KNN Graph
  - Using Locality Sensitive Hashing (LSH)
  - Using Greedy Construction: best at the moment
    Vicinity [VS05], T-Man [JMB09], NNDescent [DML11], Hyrec [BFGKP14]
Greedy KNN Construction

Parallel-iterative algorithm,

- From a random graph
- Each node looks for potential new neighbours:
  \((1)\) among random nodes
Greedy KNN Construction

Parallel-iterative algorithm,

- From a random graph
- Each node looks for potential new neighbours:
  - (1) among random nodes (optional)
  - (2) among "friends of friends"
Repeat for all users until \( \# \text{changes} < \epsilon \)

**current neighborhood**

**neighbor candidates** from (1) & (2)

**node**

**distance computation**

\[
\text{sim}(\text{node}) = \begin{bmatrix} 3 & 6 & 9 & 1 & 8 & 4 \end{bmatrix}
\]

**ranking**

**selection**

**new neighborhood**
Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- KIFF: K-nearest neighbor Fast and eFFicient
- Evaluation
Is Greed all there is?

- **Observation 1:** *Similarity* remains the **bottleneck**
  
  - 90% of execution time spent on similarity (Wikipedia dataset)

- **Observation 2:** **Datasets** are (often) **sparse**
  - Many datasets use item-based profiles
  - Most items little shared: **sparse**

F. Taiani
The Problem with Sparsity

- **Density:** \(|E| \div (|U| \times |I|)
- \(|E| = \# \text{ ratings}, \ |U| = \# \text{users}, \ |I| = \# \text{items}

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>0.7127%</td>
</tr>
<tr>
<td>Arxiv</td>
<td>0.1124%</td>
</tr>
<tr>
<td>Gowalla</td>
<td>0.0029%</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.0011%</td>
</tr>
</tbody>
</table>

- Only a few rugs ("ratings") on the ladder
  - 2 random nodes unlikely to be close
  - Hence greedy processes slow to start

*F. Taiani*
KIFF's Intuition

Greedy KNN approaches

- Assume no initial structure
- Start from a random graph

In practice

- Underlying bipartite user / item graph
- Can be used to bootstrap the greedy process
- Use items to create Ranked Candidate Sets

\[ \in RCS(? \subset ?) \text{ iff } \text{items}(? \subset ?) \cap \text{items}(\subset ?) \neq \emptyset \]
Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- KIFF: K-nearest neighbor Fast and eFFicient
- Evaluation
RCS Construction

Unrelated users are never compared
1. current neighborhood

2. top $\gamma$ candidates in $RCS_{Alice}$ by item count

$\text{sim}(A, \rightarrow) = \begin{pmatrix} C & 0.9 \\ B & 0.4 \\ D & 0.3 \\ X & 0.6 \\ Y & 0.5 \end{pmatrix}$
Indexing followed by "greedy" iteration

Trivially parallelizable + highly local

Indexing: $O(|E|)$  Iteration: $O(|U| \times |RCS|)$
Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- KIFF: K-nearest neighbor Fast and efficient
- Evaluation
Evaluation: Datasets

| Dataset     | #Users | #Items | #Ratings | Density  | Avg. $|UP_u|$ | Avg. $|IP_i|$ |
|-------------|--------|--------|----------|----------|--------|--------|
| Wikipedia   | 6,110  | 2,381  | 103,689  | 0.7127%  | 16.9   | 43.5   |
| Arxiv       | 18,772 | 18,772 | 396,160  | 0.1124%  | 21.1   | 21.1   |
| Gowalla     | 107,092| 1,280,969| 3,981,334| 0.0029%  | 37.1   | 3.1    |
| DBLP        | 715,610| 1,401,494| 11,755,605| 0.0011%  | 16.4   | 8.3    |

Long tail profile size distribution
Evaluation: Metrics

- **Wall-clock computation time**

- **Recall**
  
  - $knn_u$: *ideal* KNN neighborhood for user $u$
  
  - $\widehat{knn}_u$: *approximated* KNN neighborhood for user $u$

\[
\text{recall}_{\widehat{G}_{\text{KNN}}}(u) = \frac{|\widehat{knn}_u \cap knn_u|}{k}
\]

\[
\text{recall}(\widehat{G}_{\text{KNN}}) = \mathbb{E} \text{ recall}_{\widehat{G}_{\text{KNN}}}(u)
\]

- **Scan rate**

\[
\text{scanrate} = \frac{\#(\text{similarity evaluations})}{|U| \times (|U| - 1)/2}
\]
**Overall Performance**

<table>
<thead>
<tr>
<th>Competitor</th>
<th>speed-up</th>
<th>Δrecall</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN-Descent</td>
<td>×15.42</td>
<td>+0.14</td>
</tr>
<tr>
<td>HyRec</td>
<td>×12.51</td>
<td>+0.23</td>
</tr>
<tr>
<td>Average</td>
<td>×13.97</td>
<td>+0.19</td>
</tr>
</tbody>
</table>

*KIFF’s gain*

Arxiv

- Preprocessing
- Similarity computation
- Candidate selection

Wikipedia

- Preprocessing
- Similarity computation
- Candidate selection
Overall Performance

**KIFF’s gain**

<table>
<thead>
<tr>
<th>Competitor</th>
<th>speed-up</th>
<th>Δrecall</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN-Descent</td>
<td>×15.42</td>
<td>+0.14</td>
</tr>
<tr>
<td>HyRec</td>
<td>×12.51</td>
<td>+0.23</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>×13.97</strong></td>
<td><strong>+0.19</strong></td>
</tr>
</tbody>
</table>

---

**Gowalla**

Faster (x14), Better (+20%)
## Performance Details

<table>
<thead>
<tr>
<th>Approach</th>
<th>recall</th>
<th>wall-time (s)</th>
<th>scan rate</th>
<th>#iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arxiv</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-Descent</td>
<td>0.95</td>
<td>41.8</td>
<td>17.6%</td>
<td>9</td>
</tr>
<tr>
<td>HyRec</td>
<td>0.90</td>
<td>38.6</td>
<td>16.0%</td>
<td>12</td>
</tr>
<tr>
<td><strong>KIFF</strong></td>
<td>0.99</td>
<td>10.7</td>
<td>2.5%</td>
<td>36</td>
</tr>
<tr>
<td><strong>KIFF’s Gain</strong></td>
<td>+0.06</td>
<td>×3.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Wikipedia</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-Descent</td>
<td>0.97</td>
<td>13.1</td>
<td>51.69%</td>
<td>7</td>
</tr>
<tr>
<td>HyRec</td>
<td>0.95</td>
<td>9.4</td>
<td>44.64%</td>
<td>8</td>
</tr>
<tr>
<td><strong>KIFF</strong></td>
<td>0.99</td>
<td>4.4</td>
<td>7.37%</td>
<td>22</td>
</tr>
<tr>
<td><strong>KIFF’s Gain</strong></td>
<td>+0.03</td>
<td>×2.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Gowalla</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-Descent</td>
<td>0.69</td>
<td>307.9</td>
<td>3.67%</td>
<td>16</td>
</tr>
<tr>
<td>HyRec</td>
<td>0.56</td>
<td>253.2</td>
<td>2.69%</td>
<td>22</td>
</tr>
<tr>
<td><strong>KIFF</strong></td>
<td>0.99</td>
<td>146.6</td>
<td>0.84%</td>
<td>115</td>
</tr>
<tr>
<td><strong>KIFF’s Gain</strong></td>
<td>+0.36</td>
<td>×1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BLP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NN-Descent</td>
<td>0.78</td>
<td>10,890.2</td>
<td>3.08%</td>
<td>19</td>
</tr>
<tr>
<td>HyRec</td>
<td>0.63</td>
<td>8,829.9</td>
<td>2.37%</td>
<td>26</td>
</tr>
</tbody>
</table>

Much reduced scan rate
KIff's Scan Rate

Arxiv Dataset

KIff: First iterations yield highest gains
Impact of RCS on Bootstrap

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Top $k$ from RCS</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arxiv</td>
<td>0.82</td>
<td>0.08</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>0.54</td>
<td>0.01</td>
</tr>
<tr>
<td>Gowalla</td>
<td>0.55</td>
<td>0.15</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.79</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Recall

F. Taiani
Repeat for all users until \( \# \text{changes} \leq \beta \)

Termination Criteria

- Vertical bars: RCS truncation imposed by KIFF

Termination only impacts minority of users
# Effect of Density

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ratings</th>
<th>Density</th>
<th>average</th>
<th>RCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML-1</td>
<td>1,000,209</td>
<td>4.47%</td>
<td>2,892.7</td>
<td></td>
</tr>
<tr>
<td>ML-2</td>
<td>500,009</td>
<td>2.23%</td>
<td>2,060.6</td>
<td></td>
</tr>
<tr>
<td>ML-3</td>
<td>255,188</td>
<td>1.14%</td>
<td>1,125.4</td>
<td></td>
</tr>
<tr>
<td>ML-4</td>
<td>131,668</td>
<td>0.59%</td>
<td>510.8</td>
<td></td>
</tr>
<tr>
<td>ML-5</td>
<td>68,415</td>
<td>0.30%</td>
<td>202.5</td>
<td></td>
</tr>
</tbody>
</table>

![Wall-time (s) and Scanrate graphs for different datasets]
Effect of Density

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ratings</th>
<th>Density</th>
<th>average RCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML-1</td>
<td>1,000,209</td>
<td>4.47%</td>
<td>2,892.7</td>
</tr>
<tr>
<td>ML-2</td>
<td>500,009</td>
<td>2.23%</td>
<td>2,060.6</td>
</tr>
<tr>
<td>ML-3</td>
<td>255,188</td>
<td>1.14%</td>
<td>1,125.4</td>
</tr>
<tr>
<td>ML-4</td>
<td>131,668</td>
<td>0.59%</td>
<td>510.8</td>
</tr>
<tr>
<td>ML-5</td>
<td>68,415</td>
<td>0.30%</td>
<td>202.5</td>
</tr>
</tbody>
</table>

Scan rate grows with density, hurting perf
Conclusion

- Novel KNN construction algorithm
- Intuition: reduce similarity computations
- Faster (x14) and more accurate (+20%) than SotA
- Performs best on sparse datasets
- Future: finer complexity analysis / distributed version
Thank you
Some References


