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

François Taïani – ASAP – Inria Rennes
Ideal computer system

- structured, predictable, open, evolvable
A Distributed System Today ...

- facebook
- twitter
- bit.ly
- foursquare
  Geosocial app, est. 2009

- Standards
  - OAuth
  - JSON

- External developers
  - User

- External services
  - Fish

- Middleware
  - mongoDB
  - Flume

- 45M Users

- Amazon web services
Today's computer systems

- sprawling, chaotic, complex, unmanageable?

Sprawling

Chaotic
one RPC request,
• **2065** individual invocations
• > **50** C-functions
• > **140** C++ classes

Source: [TKF2009]
Unmanageable?

- **Globus** client
  - 1 creation, 4 requests, 1 destruction

- **Projection** w.r.t.
  - stack depth
  - package

client: **1,544,734** Java method call (sic)
server: **6,466,652** Java method calls (sic) [+time out]

The Impact of Web Service Integration on Grid Performance. Taïani, Hiltunen, Schlichting, HPDC-14, 2005
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

Netflix awarded a $1 million prize to the 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 for the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.
Which practical approaches for scale and performance?

$1 million prize

recommended

too much engineering effort
Outline

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

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

- Published at ICDE 2016
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, ...

\[
sim(Alice, Bob) = \text{profile of Alice} \sim \text{profile of Bob}
\]
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 \( \# \text{changes} < \varepsilon \)

**current neighborhood**

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

**distance computation**

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

**ranking**

**selection**

**new neighborhood**

**Greedy Procedure**
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**
The Problem with Sparsity

Density:  \( \frac{|E|}{(|U| \times |I|)} \)

\( |E| = \# \text{ ratings}, |U| = \# \text{users}, |I| = \# \text{items} \)

<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

density = 35%
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

\[ \in RCS(\text{user}) \iff \text{items(\text{user})} \cap \text{items(\text{item})} \neq \emptyset \]
The problem: KNN graph construction
The intuition: Is greed all there is?
KIFF: K-nearest neighbor Fast and eFFicient
Evaluation
RCS Construction

Users

Alice

Bob

Darth

Stormy

Items

items_{Alice}

items_{Bob}

IP_{chalet}

IP_{bank}

Users Items
Unrelated users are never compared
1. Current neighborhood

2. Top $\gamma$ candidates in $\text{RCS}_{\text{Alice}}$ by item count

$\text{RCS}_{\text{Alice}}$

- Xavier: 6
- Yann: 3
- ... 

Sorted by item count

$\text{sim}(A, -) = \begin{align*}
\text{Bob} & : 0.4 \\
\text{Dave} & : 0.9 \\
\text{C} & : 0.3 \\
\text{B} & : 0.6 \\
\text{D} & : 0.5 \\
\end{align*}$
Indexing followed by "greedy" iteration

Trivially parallelizable + highly local

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

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Users</th>
<th>U</th>
<th>#Items</th>
<th>I</th>
<th>#Ratings</th>
<th>E</th>
<th>Density</th>
<th>Avg.</th>
<th>UP_u</th>
<th>Avg.</th>
<th>IP_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wikipedia</td>
<td>6,110</td>
<td>2,381</td>
<td>103,689</td>
<td>0.7127%</td>
<td>16.9</td>
<td>43.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arxiv</td>
<td>18,772</td>
<td>18,772</td>
<td>396,160</td>
<td>0.1124%</td>
<td>21.1</td>
<td>21.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gowalla</td>
<td>107,092</td>
<td>1,280,969</td>
<td>3,981,334</td>
<td>0.0029%</td>
<td>37.1</td>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBLP</td>
<td>715,610</td>
<td>1,401,494</td>
<td>11,755,605</td>
<td>0.0011%</td>
<td>16.4</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Long tail profile size distribution
Evaluation: Metrics

- Wall-clock **computation time**

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

  $$
  \text{recall}_\hat{G}_{KNN}(u) = \frac{|\hat{knn}_u \cap knn_u|}{k}
  $$

  $$
  \text{recall}(\hat{G}_{KNN}) = \mathbb{E} \ \text{recall}_\hat{G}_{KNN}(u)
  $$

- **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><strong>Average</strong></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>Average</td>
<td>×13.97</td>
<td>+0.19</td>
</tr>
</tbody>
</table>

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>KIFF</td>
<td><strong>0.99</strong></td>
<td><strong>10.7</strong></td>
<td><strong>2.5%</strong></td>
<td><strong>36</strong></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>Gowalla</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>KIFF</td>
<td><strong>0.99</strong></td>
<td><strong>4.4</strong></td>
<td><strong>7.37%</strong></td>
<td><strong>22</strong></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>BLP</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>KIFF</td>
<td><strong>0.99</strong></td>
<td><strong>146.6</strong></td>
<td><strong>0.84%</strong></td>
<td><strong>115</strong></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>Wikipedia</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>
Termination Criteria

Repeat for all users until \( \text{changes} \leq \beta \)

Virtual 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 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>

![Graph showing wall-time (s) and scanrate for different datasets and methods]
Effect of Density

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


