Scalable, Generic, and Adaptive Systems for Focused Crawling

Georges Gouriten* - georges@netiru.fr
Silviu Maniu°
Pierre Senellart*°

* Télécom Paristech – Institut Mines-Télécom – LTCI CNRS
° Hong Kong University
What is focused crawling?
A directed graph
Web

Social network

P2P

e tc.
Weighted
Let $u$ be a node,

$$\beta(u) = \text{count of the word } Bhutan \text{ in all the tweets of } u$$
Even more weighted
Let \((u, v)\) be an edge,

\[ \alpha(u) = \text{count of the word } Bhutan \text{ in all the tweets of } u \text{ mentioning } v \]
The total graph
A seed list
The frontier
Crawling one node
A crawl sequence

Let \( V_0 \) be the seed list, a set of nodes, a "crawl sequence", starting from \( V_0 \), is

\[
\{ v_i, v_i \text{ in frontier}(V_0 \cup \{v_0, v_1, \ldots, v_{i-1}\}) \}
\]
Goal of a focused crawler

Produce crawl sequences with global scores (sum) as high as possible
A high-level algorithm

Estimate scores at the frontier
Pick a node from the frontier
Crawl the node
Supposing a perfect estimator
Finding an optimal crawl sequence offline:
NP-hard

Greedy wins for a crawled graph > 1000 nodes

Refresh rate of 1 is better
Estimation in practice
Different kinds of estimators
bfs
**Estimator 1 (bfs).** \( \tilde{\beta}(v) = \frac{1}{l(v)+1} \), where \( l(v) \) is the distance of \( v \) to \( V_0 \).
navigational rank

score propagation from the ancestors of a node

then to the children of a node
\[ NR_1(v)^{t+1} = d \times w(v) + (1 - d) \times \text{avg}_{(v,u) \in E'} \frac{NR_1(u)^t}{d_i(u)} \]

\[ NR_2(v)^{t+1} = d \times NR_1(v) + (1 - d) \times \text{avg}_{(u,v) \in E'} \frac{NR_2(u)^t}{d_o(u)}. \]

**Estimator 2 (nr).** \( \tilde{\beta}(v) = NR_2(v). \)
opic

online page importance computation

~ online pageRank computation
1. the node $v$ with the highest cash is selected, and its history is updated with the current cash value $H(v) = H(v) + C(v)$,

2. for each outgoing node $u$ of $v$, the cash value is updated $C(u) = C(u) + \frac{C(v)}{d_{o(v)}}$,

3. the cash value of $v$ is reset and the global counter incremented, by $G = G + C(v)$ and $C(v) = 0$.

$$2. \rightarrow C(u) = C(u) + \frac{C(v)}{\sum_{(v,w) \in E'} \alpha(v,w) \times C(w) \times \alpha(v,u) \times C(u)}$$

Estimator 3 (opic). $\tilde{\beta}(v) = \frac{H(v) + C(v)}{G + 1}$. 
Open spaces in the state-of-the-art

nr has a quadratic complexity

opic focus on popularity

the rest is about how to score
First-level neighborhood
Second-level neighborhood
Neighborhood-based estimators

Estimator 4 (\texttt{fl\_n fl\_e fl\_ne sl\_n sl\_e sl\_ne}).

\texttt{fl\_deg:} \quad \widetilde{\beta}(v) = d_i(v) = |P(v)|

\texttt{fl\_n:} \quad \tilde{\beta}(v) = \sum_{u \in P(v)} \beta(u)

\texttt{fl\_e:} \quad \tilde{\beta}(v) = \sum_{u \in P(v)} \alpha(u,v)

\texttt{fl\_ne:} \quad \tilde{\beta}(v) = \sum_{u \in P(v)} \beta(u) \alpha(u,v)

\texttt{sl\_n:} \quad \widetilde{\beta}(v) = \sum_{u \in P(v)} \sum_{w \in V' \atop u \in P(w)} \beta(w)

\texttt{sl\_e:} \quad \tilde{\beta}(v) = \sum_{u \in P(v)} \sum_{w \in V' \atop u \in P(w)} \alpha(u,w)

\texttt{sl\_ne:} \quad \tilde{\beta}(v) = \sum_{u \in P(v)} \sum_{w \in V' \atop u \in P(w)} \beta(w) \alpha(u,w)
deg, e, n, ne

deg: number of neighbors

e: sum of incoming edges

n: sum of incoming nodes

ne: sum of incoming (node*edge)s
Linear regressions

Estimator 5 \((lr\_fl \ lr\_sl)\).
\(lr\_fl\): \(\hat{\beta}(v) = \text{trained linear combination of the fl\_ estimators.}\)
\(lr\_sl\): \(\tilde{\beta}(v) = \text{trained linear combination of the fl\_ and sl\_ estimators.}\)
Multi-armed bandits (1)
Multi-armed bandits (2)

Budget n, how to maximize the reward?

Balance exploration and exploitation
Applied to focused crawling

Slot machines: estimators

Reward: score of the top node
mab_ε

probability 1-ε: slot machine with the highest average reward

probability ε: random slot machine

Estimator 6 (mab_ε). \( \tilde{\beta}(v) = \text{output of an epsilon-greedy strategy.} \)
mab_\(\varepsilon\)-first

steps \([0, \varepsilon \times N]\): random slot machine

steps \([\varepsilon \times N + 1, N]\): slot machine with the highest average reward

**Estimator 7** (mab_\(\varepsilon\)-first). \(\hat{\beta}(v) = \text{output of an epsilon-first strategy.}\)
mab_var

Succession of $\varepsilon$-first strategies, with a reset every $r$ steps, $r$ varying with the context

**Estimator 8** (mab_var). $\tilde{\beta}(v) =$ output of an epsilon-first with variable reset strategy.
Their running times
Expected running times

Twitter API for one week:
- 3s
- 200,000 nodes

One domain website for one week:
- 1s
- 600,000 nodes
## Experimental framework (1)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes (million)</th>
<th>Non-zero nodes (%)</th>
<th>Edges (million)</th>
<th>Non-zero edges (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRETAGNE</td>
<td>2.2</td>
<td>2.0</td>
<td>35.6</td>
<td>0.5</td>
</tr>
<tr>
<td>FRANCE</td>
<td>&quot;</td>
<td>19.2</td>
<td>&quot;</td>
<td>6.8</td>
</tr>
<tr>
<td>HAPPY</td>
<td>16.9</td>
<td>11.0</td>
<td>78.0</td>
<td>2.4</td>
</tr>
<tr>
<td>JAZZ</td>
<td>&quot;</td>
<td>0.6</td>
<td>&quot;</td>
<td>0.1</td>
</tr>
<tr>
<td>WEIRD</td>
<td>&quot;</td>
<td>3.2</td>
<td>&quot;</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Experimental framework (2)

— *Graph score*
10 seed graphs
1 seed graph:
50 seeds picked randomly among non-zero $\beta$
Arithmetic average of the crawl scores (sum)

— *Global score*
Normalization with a baseline -- *relative score*
Geometric average among the five graphs
Datasets and code are online

http://netiru.fr/research/14fc
To measure the running times

Same crawl sequence: the oracle
Storage in RAM (20G)
3.6 GHz
## The running times (ms)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Evaluator</th>
<th>100</th>
<th>1,000</th>
<th>10,000</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FRANCE</strong></td>
<td>nr</td>
<td>2,832.1</td>
<td>19,720.5</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>opic</td>
<td>1.9</td>
<td>2.5</td>
<td>4.6</td>
<td>4.7</td>
</tr>
<tr>
<td></td>
<td>ne_fl</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>lr_fl</td>
<td>0.2</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>mab_var_fl</td>
<td>0.6</td>
<td>0.3</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>ne_sl</td>
<td>8.5</td>
<td>27.1</td>
<td>2.0</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td>lr_sl</td>
<td>8.5</td>
<td>27.2</td>
<td>2.0</td>
<td>6.1</td>
</tr>
<tr>
<td><strong>HAPPY</strong></td>
<td>nr</td>
<td>45,965.7</td>
<td>105,209.3</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>opic</td>
<td>1.8</td>
<td>1.6</td>
<td>1.9</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>ne_fl</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>lr_fl</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>mab_var_fl</td>
<td>1.1</td>
<td>0.3</td>
<td>0.5</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>ne_sl</td>
<td>111.1</td>
<td>24.5</td>
<td>63.3</td>
<td>240.5</td>
</tr>
<tr>
<td></td>
<td>lr_sl</td>
<td>111.4</td>
<td>24.5</td>
<td>63.3</td>
<td>241.0</td>
</tr>
</tbody>
</table>
$$NR_1(v)^{t+1} = d \times w(v) + (1 - d) \times \text{avg}_{(v,u) \in E'} \frac{NR_1(u)^t}{d_i(u)}$$

$$NR_2(v)^{t+1} = d \times NR_1(v) + (1 - d) \times \text{avg}_{(u,v) \in E'} \frac{NR_2(u)^t}{d_0(u)}.$$

**Estimator 2 (nr).** $\tilde{\beta}(v) = NR_2(v)$.

Quadratic complexity, with large constant factors
Their precision
The precision

Same crawl sequence: the oracle

Precision: distance of the top node to the actual top node

Arithmetically averaged over a window of 1000 steps
For bretagne
Their ability to lead crawls
Leading the crawl

Different crawl sequences:

defined by the top estimated nodes
Average graph scores for France
The multi armed-bandits

<table>
<thead>
<tr>
<th>Type</th>
<th>100</th>
<th>1,000</th>
<th>10,000</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0.450</td>
<td>0.481</td>
<td>0.477</td>
<td>0.495</td>
</tr>
<tr>
<td>$\varepsilon$-first</td>
<td>0.409</td>
<td>0.501</td>
<td>0.484</td>
<td>0.490</td>
</tr>
<tr>
<td>var-0.1-1000</td>
<td>0.383</td>
<td>0.439</td>
<td>0.420</td>
<td>0.494</td>
</tr>
<tr>
<td>var-0.2-200</td>
<td>0.427</td>
<td>0.413</td>
<td>0.461</td>
<td>0.458</td>
</tr>
</tbody>
</table>
## All the estimators

<table>
<thead>
<tr>
<th>Estimator</th>
<th>100</th>
<th>1,000</th>
<th>10,000</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>bfs</td>
<td>0.147</td>
<td>0.132</td>
<td>0.130</td>
<td>0.207</td>
</tr>
<tr>
<td>opic</td>
<td>0.283</td>
<td>0.184</td>
<td>0.205</td>
<td>0.287</td>
</tr>
<tr>
<td>n</td>
<td>0.358</td>
<td>0.280</td>
<td>0.362</td>
<td>0.467</td>
</tr>
<tr>
<td>e</td>
<td>0.594</td>
<td>0.560</td>
<td>0.457</td>
<td>0.377</td>
</tr>
<tr>
<td>ne</td>
<td>0.583</td>
<td>0.570</td>
<td>0.466</td>
<td>0.378</td>
</tr>
<tr>
<td>lr_fl</td>
<td>0.325</td>
<td>0.382</td>
<td>0.466</td>
<td>0.504</td>
</tr>
<tr>
<td>mab_var-0.2-200</td>
<td>0.427</td>
<td>0.413</td>
<td>0.461</td>
<td>0.458</td>
</tr>
</tbody>
</table>
Conclusion
What we learnt

Generic model

NP-hardness offline

Refresh rate of 1
Greedy

Neighborhood features
Linear regressions
Multi-armed bandit strategy
Future work

Approximation of the optimal score

Distributed crawl

Recrawling nodes

Further multi-armed bandits comparisons
Thank you.

georges@netiru.fr
Finding the optimal crawl sequences in a known graph
PTime many-one reduction from the LST-Graph problem

Problem remains hard if nodes, not edges, are weighted

A subtree rooted at \( r \) is seen as a crawl sequence starting from \( r \)

Free edges are added to the graph to allow free crawls from he seed to any potential root of a subtree
Rich friends will make you richer
The greedy strategy

Node picked = argmax(\(\beta(v)\)), \(v\) in frontier
Is not always optimal
The altered greedy strategy

Node picked =

probability q: \( \text{argmax}(\beta(v)) \)

probability 1-q: random \( v \) so that,
\[ \max(\beta(u)) - \beta(v) \leq \zeta \times \max(\beta(u)) \]
Altered greedy vs greedy for jazz

![Graph showing relative score vs greedy budget for different parameters.]

- $q = 0.8, \zeta = 1.0$
- $q = 0.8, \zeta = 0.8$
- $q = 0.8, \zeta = 0.6$
- $q = 0.8, \zeta = 0.4$
The refresh rate disadvantage
When estimation takes too long

```
input : seed subgraph $G_0$, budget $n$
output: crawl sequence $V$ with a score as high as possible
1 $V \leftarrow ()$
2 $G' \leftarrow G_0$
3 budgetLeft $\leftarrow n$
4 while budgetLeft $> 0$ do
5     frontier $\leftarrow$ extractFrontier($G'$);
6     scoredFrontier $\leftarrow$
7         estimator.scoreFrontier($G'$, frontier);
8     $r \leftarrow$ getRefreshRate();
9     NodeSequence $\leftarrow$
10        strategy.getNextNodes(scoredFrontier, $r$);
11     $V \leftarrow (V, NodeSequence)$;
12 for $u$ in NodeSequence do
13     $G' \leftarrow G' \cup$ crawlNode($u$);
14     budgetLeft $= budgetLeft - r$
15 return $V$
```
The score degradation (%) at different steps

<table>
<thead>
<tr>
<th>Refresh rate</th>
<th>100</th>
<th>1,000</th>
<th>10,000</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.4</td>
<td>2.2</td>
<td>3.9</td>
<td>6.4</td>
</tr>
<tr>
<td>8</td>
<td>1.3</td>
<td>6.5</td>
<td>12.8</td>
<td>18.3</td>
</tr>
<tr>
<td>32</td>
<td>6.6</td>
<td>6.5</td>
<td>17.5</td>
<td>24.3</td>
</tr>
<tr>
<td>128</td>
<td>38.8</td>
<td>10.7</td>
<td>19.9</td>
<td>29.5</td>
</tr>
<tr>
<td>1024</td>
<td>38.8</td>
<td>74.3</td>
<td>25.8</td>
<td>35.9</td>
</tr>
</tbody>
</table>