Trie structure for graph label functions

Philippe Jacquet
Bell Labs
Classic trie structure on digital sequences

X2=aababba...
X3=bbabbaa...

Trie generalization for graph.

• Take an infinite acyclic graph $G = G(V,E)$ (finite degrees). Take a finite alphabet $\mathcal{A}$.

• A function $X: V \rightarrow \mathcal{A}$ is called a label function.
Trie generalization for graph.

• Take an infinite acyclic graph $G = G(V,E)$ (finite degrees). Take a finite alphabet $\mathcal{A}$.

• A function $X: V \rightarrow \mathcal{A}$ is called a label function.
Trie generalization for graph.

• Take an infinite acyclic graph $G=G(V,E)$ (finite degrees). Take a finite alphabet $\mathcal{A}$.
• A function $X: V \rightarrow \mathcal{A}$ is called a label function.
Trie generalization for graph

• How to build a G-trie from n label functions $X_1, \ldots, X_n$?
• When the graph G is the semi-infinite oriented chain:
  – Label functions are the linear sequences on $\mathcal{A}$.
  – The G-Trie is a classic trie

– Terminology collision: not to be confused with g-trie for subgraph pattern matching
  • Ribeiro, P., & Silva, F. “G-tries: an efficient data structure for discovering network motifs”. 2010
G trie construction on label functions

• We select an anchored node $O$ in $G$
• For all integers $k$, every path $P$ of length $k$ from $O$ in $G$, and for all words $w$ in $\mathcal{A}^k$.

• The node $(P, w)$ belongs to the G-Trie iff
  – At least two label functions read $w$ on path $P$.
• The G-Trie is a tree due to the tree nature of paths and words sets.
G-trie versus trie

• Restricted to a single (infinite) path $P$ the G-trie is a classic trie on alphabet $\mathcal{A}$

• If $G$ is $M$ regular, the G-trie is a $MxA$ ary tree
Example of Graph Trie

X2 = bababab... bhababa...

X3 = ababab... bababa...
Example of Graph Trie

X2 = bababab...
    bbbababa...

X3 = ababab...
    bababa...
Example of Graph Trie

X2= bababab...
    babababa...

X3= ababab...
    bababa...
Classic java trie

Class Trie{
   id index;
}
Class Node extends Trie{
   Trie[] subtries;
}
Class Leaf extends Trie{
   record content;
}

Trie insert(record Y, Node T){
   Trie[] S=T.subtries; id i=T.index; char a=Y[i+1];
   if (S[a]=null){S[a]=new Leaf(i+1,Y);}
   else {S[a]=insert(Y,S[a]);}
   return new Node(i,S);
}

Trie insert(record Y, Leaf T){
   id i=T.index; record X=T.content; char a=X[i+1];
   Trie[] S=new Trie[A];
   S[a]=new Leaf(i+1,X);
   return insert(Y,new Node(i,S));
}
Java G trie

Class GTrie { id index; }
Class GNode extends GTrie { GTrie[][] subGtries; }
Class GLeaf extends Gtrie { label content; }

Trie insert (record Y, GNode T) {
   Trie[][] S = T.subGtries; id i = T.index;
   for (j = 0, j < deg(i), j++) {
      char a = Y[next(i)[j]];
      if (S[a][j] = null) { S[a][j] = new GLeaf(next(i)[j], Y); }
      else { S[a][j] = insert(Y, S[a][j]); }
   }
   return new GNode (i, S);
}

Trie insert (record Y, GLeaf T) {
   id i = T.index; label X = T.content;
   Trie[][] S = new GTrie[A][deg(i)];
   for (j = 0, j < deg(i), j++) {
      char a = X[next(i)[j]];
      S[a][j] = new GLeaf(next(i)[j], X);
   }
   return insert(Y, new GNode(i, S));
}
Java G trie

Class GTree{ id index;}
Class GNode extends GTree{GTree[][] subGtries;}
Class GLeaf extends GTree{label content;}

Trie insert(record Y, GNode T){
    Trie[][] S=T.subGtries; id i=T.index;
    for (j=0,j<deg(i),j++)
        char a=Y[next(i)[j]];
        if (S[a][j]=null){S[a][j]=new GLeaf(next(i)[j],Y);} else {S[a][j]=insert(Y,S[a][j]);}
    return new GNode (i,S);
}

Trie insert(record Y, GLeaf T){
    id i=T.index; label X=T.content;
    Trie[][] S=new GTree[A][deg(i)];
    for (j=0,j<deg(i),j++)
        char a=X[next(i)[j]];
        S[a][j]=new GLeaf(next(i)[j],X);
    return insert(Y,new GNode(i,S));
}
Space-time pattern matching

• « Time expansion » of a static graph $G_S$
  – Expand each vertex $U$ in $G$ in a chain $U_1, U_2, \text{etc}$, to form the vertices of $G$.
  – In the expended graph $G$ edges are:
    • $(U_n, U_{n+1})$
    • $(U_n, V_{n+1})$ iff $(U, V)$ exists in $G_S$
Space-time pattern matching

- « Time expansion » of a static graph $G_s$
  - Expand each vertex $U$ in $G$ in a chain $U_1, U_2, \text{etc}$, to form the vertices of $G$.
  - In the expended graph $G$ edges are:
    - $(U_n, U_{n+1})$
    - $(U_n, V_{n+1})$ iff $(U, V)$ exists in $G_s$
Space time pattern matching

Is the time expansion of
Toward G suffix tree

• Application: to find patterns in a time expanded graph (eg alarm logs in networks)
  – Assume a unique label function
• Build the G trie from the n label functions obtained
  – By time translations of the original label function.
  – Generalized suffix tree.
G suffix tree

• On a lattice
G suffix tree

• On a lattice
G suffix tree

- On a lattice
G suffix tree

• On a lattice
Back to math

• Consider an acyclic infinite graph and an anchored node.
  – $T_k$ is the number of path of length $k$ from O.
    \[ T(z) = \sum_{k} T_k z^k \]
  – In the lattice \hspace{0.1cm} $T_k = 2^k$
    \[ T(z) = \frac{1}{1-2z} \]
Back to math

- Consider $n$ i.i.d label functions
  \[ H(s) = \sum_{a \in \mathcal{A}} p_a^{-s} \]
- $S_n$ is the average G-trie size.
- Non explosive condition
  \[ S_2 < \infty \iff T(H(-2)) < \infty \]
  - If $G$ is $M$ regular the condition becomes $MH(-2) < 1$
    - Which implies $M < A = |\mathcal{A}|$
  - If $G$ is stationary the condition becomes $\lambda(G)H(-2) < 1$
- From now we assume non explosive condition
### Summary of results

<table>
<thead>
<tr>
<th></th>
<th>Classic trie</th>
<th>G trie (non explosive)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average size</strong></td>
<td>$\Theta(n)$</td>
<td>$\Theta(n^\rho)$</td>
</tr>
<tr>
<td><strong>Size variance</strong></td>
<td>$\Theta(n)$</td>
<td>$O(n^{2\rho-1})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uniform alphabet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Theta(n^\rho)$</td>
</tr>
<tr>
<td><strong>Average insertion cost</strong></td>
<td>$\Theta(\log n)$</td>
<td>$\Theta(n^{\rho-1})$</td>
</tr>
<tr>
<td><strong>Insertion cost variance</strong></td>
<td>$O(\log n)$</td>
<td>$O(n^{\rho-1})$</td>
</tr>
<tr>
<td></td>
<td>Uniform alphabet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$O(1)$</td>
<td></td>
</tr>
<tr>
<td><strong>(Average height)</strong></td>
<td>$\Theta(\log n)$</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>
Average G-Trie size

- Poisson generating function
  \[ S(z) = \sum_{n} S_n \frac{z^n}{n!} \]
  \[ S(z) = \sum_{k} T_k \sum_{w \in \mathcal{A}^k} \left( 1 - e^{-P(w)z} - P(w)ze^{-P(w)z} \right) \]

- Mellin transform
  \[ S^*(s) = \int_{0}^{\infty} S(z)z^{s-1}dz \]
  \[ S^*(s) = (1 + s)\Gamma(s)T(H(s)) \]
  - Defined for
    \[ -2 < \Re(s) < -\rho \]
    - With \( \rho \) the first singularity of \( T(H(s)) \)
    - \( \rho > 1 \), and \( \rho < 2 \) (non explosive condition)
    - M regular graph uniform case
      \[ \rho = 1 + \frac{\log M}{\log A} \]
  - Singularities at \( \Re(s) = -\rho \) and beyond determine asymptotics
Average G-Trie size

- M regular graph: \( S(z) = M \sum_{a \in \mathcal{A}} S(p_au) + 1 - (1 + z)e^{-z} \)

\[
S^*(s) = (1 + s)\Gamma(s) \frac{1}{1 - MH(s)}
\]

\( MH(-\rho) = 1 \)

- Easy theorem: *We have the expansion*

\[
S(z) = \frac{(1 - \rho)\Gamma(-\rho)}{MH'(\rho)} z^\rho + \sum_k B_k z^{\beta_k} + O(z^{\beta'}) \quad \text{with } \text{Re}(\beta_k) \leq \rho \text{ and } \beta' < \rho
\]

- Depoissonization:

\[
S_n = S(n) + O(n^{\rho-1})
\]
Insertion cost

• \( C_n \) the average number of visited or newly created nodes in G trie
  – when inserting \( X_{n+1} \)

\[
C(z) = M \sum_{a \in \mathcal{A}} p_a C(p_a z) + 1 - e^{-z}
\]

• Theorem
\[
C_n = \frac{\Gamma(1-\rho)}{MH'(-\rho)} n^{\rho-1} + \sum_k D_k z^{\beta_k} + O(n^{\rho-1})
\]
Insertion cost

• $C_n$ the average number of visited or newly created nodes in G trie
  – when inserting $X_{n+1}$

\[ C(z) = M \sum_{a \in \mathcal{A}} p_a C(p_a z) + 1 - e^{-z} \]

• Theorem

\[ C_n = \frac{\Gamma(1 - \rho)}{MH'(-\rho)} n^{\rho-1} + \sum_k D_k z^\beta_k + O(n^{\rho-1}) \]
Hadamard product of Poisson generating functions


\[ f(z) = \sum_n f_n \frac{z^n}{n!} e^{-z} \quad g(z) = \sum_n g_n \frac{z^n}{n!} e^{-z} \]

\[ (f \ast g)(z) = \sum_n f_n g_n \frac{z^n}{n!} e^{-z} = f(z) \ast g(z) \]

• Under general depoissonization conditions (see end)

\[ (f \ast g)(z) = f(z) g(z) + zf'(z)g'(z) + O(z^{\beta_1 + \beta_2 - 2}) \]

when \( f(z) = O(z^{\beta_1}) \) and \( g(z) = O(z^{\beta_2}) \)
Size variance

- When $G$ is a M-ary tree (branches are independent)

  - Poisson variance
    \[
    V(z) = M \sum_{a \in \mathcal{A}} V(p_a z) - 2S(z)(1 + z)e^{-z} + (1 + z)^2 e^{-2z} - (1 + z)e^{-z} \\
    + M(M - 1) \left( \left( \sum_{a \in \mathcal{A}} S(p_a z) \right)^2 - \left( \sum_{a \in \mathcal{A}} S(p_a z) \right)^2 \right)
    \]
  - Using the fact that
    \[
    \left( \sum_{a \in \mathcal{A}} S(p_a z) \right)^2 - \left( \sum_{a \in \mathcal{A}} S(p_a z) \right)^2 = z \left( \sum_{a \in \mathcal{A}} p_a S'(p_a z) \right)^2 + O(z^{2 \rho - 2})
    \]
  - We get
    \[
    V(z) = O(z^{2 \rho - 1})
    \]
Size variance

• With depoissonization: \[ V_n = V(n) - n(S'(n))^2 + O(n^{2\rho-2}) \]

• Notice that \(2\rho-1 > \rho\)
  
  – the variance is of larger order than the mean


  \[ p_a = 1/A \]

• Uniform case, and only in this case, systematic coefficient cancelation leads to

  \[ V_n = O(n^\rho) \]

  – Thanks to an anonymous referee
Insertion cost variance

- Poisson variance $W(z)$ is of order $z^{\rho-1}$
- Variance $W_n = O(n^{\rho-1})$
  - Same order than mean
  - Gaussian limit from Neininger Rüschendorf.
Limiting distribution

• Size pgf \( P(z,u) = \sum_{n,k} P(S_n = k) u^k \frac{z^n}{n!} \)

\[
P(z,u) = u \left( \prod_{a \in \mathcal{A}} P(p_a z, u) \right)^{\ast M} + (1-u)(1+z)e^{-z}
\]

• Insertion cost pgf \( D(z,u) = \sum_{n,k} P(C_n = k) u^k \frac{z^n}{n!} \)

\[
D(z,u) = u \left( \prod_{a \in \mathcal{A}} p_a D(p_a z, u) \right)^{\ast M} + (1-u)e^{-z}
\]
A last theorem about dePoissonization

\[ f(z) = \sum_{n} f_n \frac{z^n}{n!} e^{-z} \]

- JS condition: \( \exists C, \beta, \alpha < 1 \)

  - in \( z \in C \Rightarrow f(z) = O(|z|^{\beta}) \)
  - Out \( z \notin C \Rightarrow f(z) e^{-z} = O(e^{\alpha|z|}) \)

- If JS condition then depoissonization theorem applies:
  \[ f_n = f(n) - \frac{n}{2} f''(n) + O(n^{\beta-2}) \cdots \]


- JS condition is strictly equivalent to \( f_n = a(n) \)

  - With a(x) analytic and \( O(|x|^{\beta}) \) in a cone \( C' \)
    \[ f(z) = a(z) + \frac{z}{2} a''(z) + O(z^{\beta-2}) \cdots \]

  - Hadamard Poisson and other results just a consequence

Thanks questions?