A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments

Virendra C. Bhavsar*
Harold Boley**
Lu Yang*

*Faculty of Computer Science, Univ. of New Brunswick, Fredericton
**Institute for Information Technology – e-Business, NRC, Fredericton
Outline

- Introduction
- Multi-agent system architecture
- Tree representation
- Similarity of trees
- Experimental results
- Conclusion
Introduction

- Semantic Web and Web Services.
  - Virtual marketplace.
  - Buyer-seller message exchange.
- Semantic match-making in multiagent systems.
  - Keywords/keyphrases.
Multi-agent system architecture

Agent-based Community Oriented Routing Network (ACORN)
Tree representation

Why tree representation?
- Flexibly represent complex structures.
- Why arc-labelled, arc-weighted tree?

```
Car
  /    \
Ford  Explorer
     \   /
       2002
```

```
Car
  /    \
Ford  Explorer
       /  \n     Model  Year
     0.2   0.5
       /  \  \
     Make  Ford
     0.3
```

```
Matchmaking of agents

- Match-making in the Cafe.

- Programming in Java
  - Credit: 0.2
  - Duration: 2 months
  - Tuition: $1200

- Introduction to Java
  - Credit: 0.2
  - Duration: 2 months
  - Tuition: $1500

- Thinking in Java
  - Credit: 0.1
  - Duration: 3 months
  - Tuition: $1800

- Textbook
  - Credit: 0.3
  - Tuition: $1200

- Cafe
  - Carried by
  - Course 1
  - Course 2
  - Leaner 1
  - Leaner 2
  - Leaner n

- Credit
  - 0.2
  - 0.1
  - 0.3
  - 0.4
Tree representation - lexicographic order

- The arcs are labelled in lexicographic (alphabetical) order.

A tree describing “Car”.

A tree describing “Hotel”.
The depth and breadth of any subtree are not limited.

A tree that describes “Jobbank”.

Tree representation - depth and breadth

- The depth and breadth of any subtree are not limited.
Serialization of trees

- Weighted Object-Oriented RuleML.
- XML attributes for arc labels and weights.

<cterm>
  <_opc><ctor>Car</ctor></_opc>
  <_r n="Make" w="0.3"><ind>Ford</ind></_r>
  <_r n="Model" w="0.2"><ind>Explorer</ind></_r>
  <_r n="Year" w="0.5"><ind>1999</ind></_r>
</cterm>

Tree serialization in OO RuleML.

cterm[ -opc[ctor[car]],
       -r[make,w[0.3]][ind[ford]],
       -r[model,w[0.2]][ind[explorer]],
       -r[year,w[0.5]][ind[1999]]
   ]

Tree representation in Relfun.
Similarity of trees – simple trees

tree $t$

Car

Make 0.3
Ford

Year 0.7
2002

tree $t'$

Car (House)

Make 0.3
Ford

Year 0.7
1999

1
0
\[ \sum s_i (w_i + w'_i)/2 \rightarrow \sum A(s_i)(w_i + w'_i)/2 \]

\[ A(s_i) = s_i \quad \text{and} \quad A(s_i) = \sqrt{s_i} \]
Algorithm for tree similarity

- Three main recursive functions of the algorithm.
  - Treesim(t,t'): Recursively compares any (unordered) pair of trees.
  - Treemap(l,l'): Co-recursively maps two lists, l and l', of labelled and weighted arcs: descends into identical–labelled subtrees.
  - Treeplicity(i,t): Decreases the similarity with decreasing simplicity.
## Experimental results – simple trees

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Tree</th>
<th>Tree</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image" alt="Tree" /></td>
<td><img src="image" alt="Tree" /></td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Tree" /></td>
<td><img src="image" alt="Tree" /></td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Tree" /></td>
<td><img src="image" alt="Tree" /></td>
<td>1.0</td>
</tr>
</tbody>
</table>
**Experimental results – simple trees (cont’d)**

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Tree</th>
<th>Tree</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td><img src="image1" alt="Tree 1" /></td>
<td><img src="image2" alt="Tree 2" /></td>
<td>0.2823</td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="Tree 3" /></td>
<td><img src="image4" alt="Tree 4" /></td>
<td>0.1203</td>
</tr>
</tbody>
</table>
## Experimental results – identical tree structures

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Tree</th>
<th>Tree</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Tree" /></td>
<td><img src="image2" alt="Tree" /></td>
<td>0.55</td>
</tr>
<tr>
<td>4</td>
<td><img src="image3" alt="Tree" /></td>
<td><img src="image4" alt="Tree" /></td>
<td>0.7000</td>
</tr>
</tbody>
</table>
Experimental results – progressively complex trees

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Tree</th>
<th>Tree</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>auto</td>
<td><code>t_2</code></td>
<td>0.3025</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>auto</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>make 1.0</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>ford</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>t_1</code></td>
<td><code>t_3</code></td>
<td>0.3025</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>auto</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>make 0.5</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>model 0.5</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>ford</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>explorer</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>t_4</code></td>
<td><code>auto</code></td>
<td>0.3025</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>make 0.3</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>model 0.2</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>year 0.5</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>ford</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>explorer</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>2002</code></td>
<td></td>
</tr>
</tbody>
</table>
Experimental results – complex trees

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Tree</th>
<th>Tree</th>
<th>$S_i$</th>
<th>$\sqrt{S_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td><img src="" alt="Tree t1" /></td>
<td><img src="" alt="Tree t2" /></td>
<td>0.5950</td>
<td>0.7611</td>
</tr>
</tbody>
</table>
Experimental results – complex trees (cont’d)

<table>
<thead>
<tr>
<th>Experiments</th>
<th>Tree</th>
<th>Tree</th>
<th>$S_i$</th>
<th>$\sqrt{S_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><img src="image1" alt="Tree t1" /></td>
<td><img src="image2" alt="Tree t2" /></td>
<td>0.5894</td>
<td>0.6816</td>
</tr>
</tbody>
</table>
Conclusion

- Matchmaking in multiagent systems – a versatile tree similarity algorithm is proposed.
- Executable specification available in functional-logic language: Relfun.
  - A Java implementation is in progress.
- Future work - Clustering of agents.
Thank you!