Linear Average Time Extraction of Phrase-structure Fragments

Andreas van Cranenburgh

Huygens ING
Royal Netherlands Academy of Arts and Sciences

Institute for Logic, Language and Computation
University of Amsterdam

January 17, 2014
Overview

- Given a pair of trees, we can extract their overlapping fragments (compare Longest Common Subsequence of strings)
- When applied to a treebank, this yields a set of recurring patterns
- Fragments can be seen as building blocks of the treebank
Applications

▶ Statistical parsing: Sangati & Zuidema (2011)
  ⇒ Use fragments as a tree-substitution grammar
    (Data-Oriented Parsing; DOP)

▶ Stylometry, e.g., authorship attribution
  ⇒ Use fragments as features to recognize
    the style of an author

▶ Research into linguistic constructions,
  Multi-word Expressions (MWE)

van Cranenburgh (2012). Literary authorship attribution (... ) fragments
Contributions

- Complexity of the previously available algorithm is quadratic in the number of nodes in the treebank.
- The present implementation works in linear average time.
- and supports treebanks with discontinuous constituents.

Sangati et al. (2010). Efficiently extract recurring fragments from large treebanks.
Definition: tree fragment

- A tree can be seen as a sequence of productions
- A tree fragment is a connected subsequence of productions from a tree
Tree kernels

Given a pair of trees, return multiset of matching nodes

Pseudocode of Quadratic Tree Kernel (QTK):

▶ For each node of tree $a$
  ▶ For each node of tree $b$
    ▶ Are the productions of the node pair equivalent?

Collins & Duffy (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron
The fast tree kernel (FTK)

Most of these comparisons can be avoided by applying a preprocessing step:

- Sort the nodes of trees by the productions they contain (for some arbitrarily defined ordering)
- Exploit this ordering in a set intersection; i.e., loop over nodes in a and b, move to next node of a as soon as $a_i < b_j$

Moschitti (2006), Making tree kernels practical for natural lang. learning
Maximal subsets

Turn bitset of matching nodes into a representation of the tree fragment:

- Traverse tree in depth-first order
- For each matching node, extract a fragment, and don’t use its node for other fragments
- Resulting fragments are maximal and connected subgraphs
It is useful to know the occurrence frequency of the extracted fragments

- Index treebank by productions; i.e., we can obtain the set of all trees with production $A \rightarrow B C$
- For a given fragment, take intersection of trees with the productions in that fragment
- Exhaustively scan the resulting candidate trees for occurrences of the fragment
Several treebanks contain discontinuous constituents as part of their annotation (e.g., Alpino / Lassy treebank).

Using some pre- and postprocessing such trees can be supported:

Pre: Replace leaves with indices, apply canonical order to leaves
Post: Canonicalize indices in fragments

van Cranenburgh (2013), Discontinuous Parsing with an Efficient and Accurate DOP Model
Implementation

- Cython: superset of Python, translated to C code
- Trees represented as arrays of node structs, labels mapped to integers
- Fragments represented as bitsets of trees, bitset operations using macros
- Fragment extraction with (mostly) native code, Python for gluing things together (multiprocessing)
### Benchmark

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Time (hr:min)</th>
<th>fragments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sangati et al. (2010), qtk, WSJ</strong></td>
<td>160:00 10:00</td>
<td>1,023,092</td>
</tr>
<tr>
<td><strong>This work, qtk, WSJ</strong></td>
<td>93:00 6:15</td>
<td>1,032,568</td>
</tr>
<tr>
<td><strong>This work, ftk, WSJ</strong></td>
<td>2:18 0:09</td>
<td>1,023,880</td>
</tr>
</tbody>
</table>

**Table: Extracting fragments from WSJ treebank**

- Training section, binarized with $h = 1$, $v = 2$ markovization
- Work is divided over 16 cores

Sangati et al. (2010). Efficiently extract recurring fragments from large treebanks
The graph shows the relationship between treebank size and the number of fragments, as well as the time taken. The number of fragments increases linearly with the treebank size, as does the time taken. The graph includes two lines: one for fragments (yellow) and one for time (red).
Conclusion

- Fragment extraction now 70 times faster! i.e., a treebank 70 times larger than WSJ is now feasible
  - More efficient implementation (2×)
  - Algorithmic speedup (35×)
- Publicly available implementation; Cf. https://github.com/andreasvc/disco-dop