Treebanks, Trees, Querying, QC, etc.

Seth Kulick
Linguistic Data Consortium
University of Pennsylvania
skulick@ldc.upenn.edu
Outline

◆ GURT 2010/TLT 2009 talk
  ● Elementary Tree decomposition of Arabic Treebank
  ● Search over Parallel trees for Parser Analysis, Inter-annotator agreement

◆ More recent stuff
  ● Elementary Tree Decomposition of PPCEME
  ● Search over Elementary Trees with lexical constraints
  ● Search over Derivation Tree

◆ Other stuff we want to do - maybe some projects
A Quantitative Analysis of Syntactic Constructions in the Arabic Treebank

Seth Kulick, Ann Bies, Mohamed Maamouri
Linguistic Data Consortium
University of Pennsylvania
{skulick,bies,maamouri}@ldc.upenn.edu
Overview

◆ Arabic Treebank (ATB3)
  ● Morphological and syntactic annotations
  ● Trees can get big

◆ Problems in Treebank construction and use
  ● Look for linguistic structures of interest
  ● Want to compare two sets of trees
    ▪ Two or more annotators, or Parser/Gold

◆ Solution: Decompose ATB into smaller trees
  ● Based on Tree Adjoining Grammar
The march occurred in another camp for refugees north of Gaza.
Our approach

- Decompose the trees into core syntactic units
  - Inspired by Tree Adjoining Grammar
  - Modification and recursion factored out from main syntactic relations
  - Idafa structures kept as single elementary trees.

- Two key properties:
  - Resulting “elementary trees” are the meaningful syntactic units to search and score upon
  - Elementary trees are “anchored” by $\geq 1$ word(s). We exploit this for the searching and scoring.
Occur the-march the-another in camp Jabaliya for refugees north Gaza
Database of Elementary Trees

◆ Each elementary tree instance consists of:
  ● “template” – the tree structure
  ● token “anchors” in this instance

◆ Treebank “database normalization” of a sort.
  ● Repeated templates are extracted out and represented by integers in a MySQL database.
  ● Saves space, makes search faster

◆ Massive structural repetition in (a) treebank:
  ● ATB3: 402,292 tokens, 321,404 etree instances
  ● 2805 etree templates
Searching Treebank for Structures

- Idafa complexity
  - Frequency of different idafa levels

<table>
<thead>
<tr>
<th>Idafa level</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>32945</td>
</tr>
<tr>
<td>3</td>
<td>7731</td>
</tr>
<tr>
<td>4</td>
<td>1283</td>
</tr>
<tr>
<td>5</td>
<td>204</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

- Verb frames – SVO,VSO, etc. – under study
- Search across pairs of trees, for comparison of this and more complex verbal structure and idafa info
Comparing Two Sets of Trees on the Same Text

- Can arise in two ways:
  - Two or more annotators: QC for corpus construction
  - Parser/Gold: to analyze parser errors

- ... but in a meaningful way
  - Want measures based on same syntactic units annotators make decisions on
  - Want correlation between two sets of trees on same data

- Unsatisfied with existing scoring and search programs (evalb, etc.)
Parallel Annotation – How to Score?

GOLD:

PARSED:

Gold has 2-level idafa, parsed has 3-level. Gold has NP-LOC, parsed doesn’t.
Query Examples

◆ Queries are specified over etree templates
  1. (NP-LOC A$) – anchor projects to a NP-LOC
  2. (NP-DIR A$) – anchor projects to a NP-DIR
  3. (NP-ADV A$) – anchor projects to a NP-ADV
  4. (NP A (NP A (NP A$))) – a three-level idafa

◆ Queries grouped into confusion matrices
  Queries 1, 2, 3
  Query 4 by itself, showing presence or absence

◆ Each token is associated with query results for the etree instance that token anchors
Query Processing

- The etree templates are searched to determine which (likely >1) match query.
- Etree instances are searched for which take a template that satisfies query.
- Since each word is linked to an etree instance, its status for a query is now easy to determine.
- Corresponding words in the two sets of trees are gone through in sequential order, each pair checked for which queries they satisfy in their set of trees.
Query Results - 1

- 4. (NP A (NP A (NP A$))) – a three-level idafa

<table>
<thead>
<tr>
<th>Gold\Parsed</th>
<th>N</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td>194</td>
</tr>
<tr>
<td>Query 4</td>
<td>88</td>
<td>677</td>
</tr>
</tbody>
</table>

- “N” indicates absence of query 4
- (4,4) – 677 cases where trees agree on 3-level idafa
- (N,4) – 194 cases where gold does not have query 4 for a token, but parse tree does
Parallel Annotation – How to Score?

Gold etree instance for token does not have 3-level idafa, but parsed tree does.

Entry in (N,4) cell for query 4.

Gold etree instance for token does not have 3-level idafa, but parsed tree does.
Query Results - 2

1. (NP-LOC A$) – anchor projects to a NP-LOC
2. (NP-DIR A$) – anchor projects to a NP-DIR
3. (NP-ADV A$) - anchor projects to a NP-ADV

<table>
<thead>
<tr>
<th>Gold\Parsed</th>
<th>N</th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>6</td>
<td>0</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Query 1</td>
<td>16</td>
<td>28</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Query 2</td>
<td>215</td>
<td>2</td>
<td>0</td>
<td>313</td>
</tr>
<tr>
<td>Query 3</td>
<td>234</td>
<td>36</td>
<td>1</td>
<td>453</td>
</tr>
</tbody>
</table>

(1,N) – 16 cases in which gold tree token projects to NP-LOC while parsed tree does not project to NP-LOC,NP-DIR,NP-ADV
Parallel Annotation – How to Score?

**GOLD:**

Entry in (1,N) cell for query set (1,2,3)
Gold etree instance for token has NP-LOC, but parsed tree does not.
Conclusions and Future Work

- Can search over small tree fragments in entire treebank or in two sets of trees on same data
- Currently used for parser analysis Gold/Parsed, plan to use for InterAnnotator Agreement.
- Focus of ongoing work:
  - Now have searches over “derivation tree”, which encodes how the etree instances combine together
  - For example, for attachment issues – where annotators agree on the basic syntactic units, but not on how they are combined
More Recent Stuff

- Now working with PPCEME – Penn-Helsinki Parsed Corpus of Early Modern English
  - 1.8 Million Words – better speed test than PTB
  - Use of Corpus Search – better test for queries
- Elementary tree searches with lexical constraints
- Derivation tree searches over etrees.
- First speed results
PPCEME Tree Decomposition

- 1,876,675 tokens
- 1,743,932 etree instances (some have multiple anchors)
- 6025 templates
  - but not including all function tags yet
  - Function tags in PPCEME very different than PTB
Example of Tree Decomposition in PPCEME

(IP-MAT (CONJ <0>And)
 (NP-OB1 (PRO <1>he))
 (, <2>,)
 (NP-SBJ (PRO$ <3>my)
   (N <4>husband))
 (ADVP (ADVS <5>best))
 (PP (P <6>of)
   (NP (Q <7>all)))
 (VBP <8>affects))

EXTRACTED ETREE INSTANCES:

<table>
<thead>
<tr>
<th>#</th>
<th>ANCHOR</th>
<th>ETREE TEMPLATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&lt;0&gt; CONJ And</td>
<td>A1</td>
</tr>
<tr>
<td>1</td>
<td>&lt;1&gt; PRO he</td>
<td>(NP A1)</td>
</tr>
<tr>
<td>2</td>
<td>&lt;2&gt; , ,</td>
<td>A1</td>
</tr>
<tr>
<td>3</td>
<td>&lt;3&gt; PRO$ my</td>
<td>A1</td>
</tr>
<tr>
<td>4</td>
<td>&lt;4&gt; N husband</td>
<td>(NP A1)</td>
</tr>
<tr>
<td>5</td>
<td>&lt;5&gt; ADVS best</td>
<td>(ADVP A1)</td>
</tr>
<tr>
<td>6</td>
<td>&lt;6&gt; P of</td>
<td>(PP A1 NP^)</td>
</tr>
<tr>
<td>7</td>
<td>&lt;7&gt; Q all</td>
<td>(NP A1)</td>
</tr>
<tr>
<td>8</td>
<td>&lt;8&gt; VBP affects</td>
<td>(IP NP[t]-OB1^ NP[t]-SBJ^ A1)</td>
</tr>
</tbody>
</table>
Derivation Tree
Another Example Tree Decomposition

```
(IP-MAT
   (ADVP-LOC (ADV <0>here))
   (BEP <1>’s)
   (NP-SBJ (D <2>a)
   (N <3>fellow)
   (CP-REL
       (WNP-1 (<NONE- 0))
       (C (<NONE- 0))
       (IP-SUB
           (NP-SBJ (<NONE-*T*-1))
           (VBP <4>frights)
           (NP-OBJ1 (NPR <5>English))
           (PP (RP <6>out)
                (P <7>of)
                (NP (PRO$ <8>his)
                     (NS <9>wits)))))
```
And Derivation Tree

```
NP
  fellow

A:1.1,1,1

#3

A:1.1,r,1

#2

a

CP[b]-REL

WNP-1  C
  0  0

IP

NP[t]-SBJ  frights  NP[t]-OB1^ *T*-1
```
Etrees and Dtrees Queries

Lexical restrictions:

L1: pos=PRO

Etrees queries:

E1) [ (IP,)
     (NP[Root],
      (A:{lex:L1}))
     (NP[t]-OB1\{(dta:1) < NP[t]-SBJ\})]

E2) [ (NP{ROOT},
     (WNP:{empty},
      C:{empty}))
     (A:{lex:L1}))]

E3) [ (CP{ROOT},
     (!WNP, !C))
     (A1:{dta:2}))

E4) [ (CP{ROOT},
     (A1:{dta:2}))
     (E5:{dta:2}))

E5) [ (NP{ROOT},
     (E5:{dta:2}))
     (E3:{dta:2}))]

Dtrees queries:

D1) (sub{dta:1} E1 E2) # topicalized clause with PRO for NP-OB1
D2) E3 # CP with empty WNP and empty C
D3) E4 # CP without WNP or C
D4) (adj{dta:2} E5 E3) # CP with empty WNP, C sister-adjointed to head of NP
Query Steps

◆ Etree Template Search
  ● Finds (etree query, etree template) matches
  ● Stores lexical restriction or dta (derivation tree address) info for each match

◆ Example:
  ● query: (IP, (NP[t]-OB1^ {dta:1} NP[t]-SBJ^))
  ● Templates:
    (IP NP[t]-OB1^ NP[t]-SBJ^ A1)
    (CP (C (-NONE- 0)) (IP (NP[t]-OB1^ NP[t]-SBJ^ A1))
  ● For first, stores address 1.1, for second 1.2.1
Derivation Tree Search

- Starts at root of Dtree query
- Find all etree instances that satisfy etree query at root of dtree query. Uses (etree query id, template) info plus check for lexical constraints.
- Find etree instance children of each such etree instance that substitutes or adjoins in the derivation tree at the {dta} address for the parent template
Some Speed Results

- All dtree queries of just a single etree query with no lexical constraints extremely fast
- With lexical constraints, still very fast
- Some derivation tree searches still very fast
  (sub{dta:1} E1 E2)
  NP tree headed by PRO substituting into NP[t]-OB1^ 
- But this one is not: (adj{dta:2} E5 E3)
  CP with empty WHNP,C adjoining at head of NP
- Reason: finds all etrees with head of NP first! 500K instances. Need to flip order.
Other Stuff we Want to Do – 1

◆ Issues with Tree Extraction
  ● Representation of empty categories across trees?
  ● Where is standard adjunction needed?
  ● Complete implementation of function tags
  ● Verify correctness – put back together

◆ Parse with extracted trees already
  ● Issue with multiple anchor trees?
Other Stuff we Want to Do – 2

◆ More query testing – parallel or not
  ● Compare against full use of corpus search
  ● Complexity of searching on derivation tree – ever need more than two nodes? Yes, but how often?

◆ Treebank Verification –
  ● Well-formedness of elementary trees (Xia, 2000)
  ● How elementary trees combine

◆ Treebank construction using TAG

◆ Make query search really fast, put on web
Example of Tree Decomposition

S

PP-TMP
In July 2005

NP-SBJ
Turkey

VP
had

VP
signed

NP
A protocol

had

signed
Example of Tree Decomposition

One-level relations (as in Collins 2003) can be helpful but not sufficient.

In July 2005 Turkey had signed A protocol
Example of Tree Decomposition

In July 2005 Turkey signed a protocol.
Example of One Etree Instance

Template with two substitution nodes.

This template occurs many times, but with different anchors.
Test Case

◆ Test case is the Penn Arabic Treebank
  ● Continuously under construction
  ● We work on it – really want something automated

◆ Using gold/parser files
  ● More files available right now
  ● Same basic problem as Inter-Annotator Agreement

◆ Linguistic Decisions on Tree Extraction
  ● “idafa” construction as etree templates
    (NP NOUN (NP NOUN)) 2-level idafa
    (NP NOUN (NP NOUN (NP NOUN (NP NOUN)))) 3-level idafa
◆ Blah

◆ Long history for NLP, massively underutilized for corpus creation (Xia, 2001)
Arabic Treebank Example

5, 6 and 9, 10 are idafas, extracted as a single etree

<5> muxay~am refugee camp
<6> jabAliyA Jabaliya
<7> li for/to
<8> lAji}+inya refugees
<9> $amAl North
<10> gaz~+ap Gaza