Data-oriented Parsing with Lexicalized Tree Insertion Grammars

Günter Neumann
LT-lab, DFKI Saarbrücken
Two Topics

- Exploring HPSG-treebanks for Probabilistic Parsing: HPSG2LTIG
  - completed work

- Exploring Multilingual Dependency Grammars for LTIG parsing
  - work in progress
Exploring HPSG-treebanks for Probabilistic Parsing: HPSG2LTIG

• joined work with Berthold Crysmann (currently at Uni. Bonn)

• to appear as
  • Günter Neumann and Berthold Crysmann
Motivation

- Grammar compilation or approximation well-established technique for improving performance of Unification-based Grammars, such as HPSG
  - Kasper et al. (1995) propose compilation of HPSG into Tree-adjoining grammar
  - Kiefer & Krieger (2000) have derived CFG from the LinGO ERG via fixpoint computation
  - Currently no successful compilation of German HPSG into CFG
Motivation

- Corpus-based specialisation of a general grammar,
  - efficiency
  - domain adaptation
  - e.g., Samuelsson, 1994; Rayner & Carter, 1996; Neumann, 1994; Krieger, 2005; Neumann & Flickinger, 2002
Stochastic Lexicalised Tree Grammars

  - Data-driven method
  - Parse trees from original grammar are decomposed into subtrees
  - Decomposition guided by HPSG's head feature principle
  - Result is Stochastic Lexicalised Tree Substitution Grammar (no recursive adjunction)
  - Speed-up: factor 3 (including replay of unifications)
Factorisation of modification

- proposed in context of TAG induction from treebanks, e.g., Hwa (1998); Neumann (1998); Xia (1999); Chen & Shanker (2000); Chiang (2000);
  - task: reconstruct TAG derivation from CF tree
  - treebank are heuristically and manually extended with the notions of head, argument, and adjunct
Lexicalised Tree Insertion Grammars (LTIG)

- LTIG Schabes & Waters, (1995) is a restricted form of LTAG, where
  - auxiliary trees are only left- or right-adjoining, no wrapping
  - no right-adjunction to nodes created by left-adjunction is allowed, and, vice versa
  - Generative power of LTIG is context-free
Stochastic LTIG

- Initial trees with root $\alpha$
  - sum($\alpha$): $P_i(\alpha) = 1$
- Substitution
  - sum($\alpha$): $P_s(\alpha | \eta) = 1$
- Adjunction of left/right auxtrees with root $\beta$
  - sum($\beta$): $P_a(\beta | \eta) + P_a(\text{NONE} | \eta) = 1$
DFKI German HPSG Treebank

- Large-scale competence grammar of German
  - Initially developed in Verbmobil by Müller & Kasper (2000)
  - Ported to LKB (Copestake, 2001) and PET (Callmeier, 2000) platforms by Müller
  - Since 2002, major improvements by Crysmann (2003, 2005)

- Initial HPSG-treebanking effort Eiche
  - based on Redwoods-technology (Oepen et al. 2002)
  - treebank based on a subset of German Verbmobil corpus
Challenges for German: Scrambling

- Almost free permutation of arguments in clausal syntax

- Interspersal of modifiers anywhere between arguments
Challenges for German: Complex predicates

- Complex predicate formation in verb cluster
- Permutation of arguments from different verbs
Challenges for German: Verb „movement“

- Variable position of finite verb
  - V1/V2 in matrix clauses
  - V-final in embedded clauses

- Initial verb related to final cluster by verb movement
Challenges for German: Discontinuous complex predicates

- Complex predicates may be discontinuous
- Argument structure only partially known during parsing
  - Number of upstairs arguments
  - Position of upstairs arguments (shuffle)
German HPSG: Overview

- German HPSG highly lexicalised
  - Information about combinatorial potential mainly encoded at lexical level
  - Syntactic composition performed by general rule schemata
- Grammar version Aug 2004
  - 87 phrase structure rules (unary & binary)
  - 56 lexical rules + 213 inflectional rules
  - over 280 parameterised lexical leaf types
    - parameters for verbs include selection for complement case, form of preposition, verb particles, auxiliary type etc.
    - nominal parameters include inherent gender
  - over 35,000 lexical entries
Rule backbone

- Rule schemata define CF-backbone
- Rule labels represent composition principles
  - (encoded as TFS), e.g., h-comp, h-subj, h-adjunct
- No segregation of dominance and precedence:
  - grammar defines both head-initial and head-final variant of basic schemata, e.g., h-comp and comp-h
- Argument composition & scrambling
  - lexical permutation of subcat lists
  - shuffle of upstairs and downstairs complements, e.g., vcomp-h-0 ... vcomp-h-4
- Movement
  - Fronting implemented as slash percolation
  - Verb movement
Eiche treebank

- Automatic annotation of in-coverage sentences by HPSG-parser
- Manual selection of best parse with Redwoods-tools
- Treebank built on subset of Verbmobil corpus
  - average sentence length (in coverage): 7.9
  - distinct trees: 16.1
  - only unique sentence strings included
    - minimise annotation effort
    - low redundancy
Eiche treebank

- Rule backbone constitutes primary treebank data
  Full HPSG-analysis can be reconstructed deterministically
- Secondary tree representation with conventional node labels
  - encodes salient information represented in AVM associated with each node (e.g., category, slash, case, number)
  - isomorphic to derivation tree
Extraction method

- Experiment based on David Chiang's TIG parser, Chiang (2000)
- Classification of rules and rule daughters according to head, argument, or modifier status (cf. Magerman, 1995)
- HPSG2LTIG Conversion (following, Chiang):
  - Adjunct daughters (adjunction) excise tree below adjunct to form a initial adjoined tree
  - Argument daughters (substitution) excise tree below argument daughter to form initial tree, leaving behind a substitution node
  - Auxiliary trees
Extraction method

- Classification according to head, argument, or modifier status straightforward and transparent
  - treebank rooted in a rich declarative grammar
  - close correspondence of relevant distinctions to HPSG composition principles
  - no heuristics (or "recovery" of linguistic theory)
- Specification based on rule-backbone
- Automatic expansion with secondary labels
  - derivation trees
    - fold isomorphic trees into one
  - head rules and argument rules
    - expand conversion rules defined on backbone by secondary labels found in treebank
Experiment 1

- 10-fold cross-validation over 3528 sentences from Verbmobil corpus
- Anchors of extracted trees (LEX) are highly specific preterminals including POS information, morphosyntax (case, number, gender, person, tense, mood), valency etc.
- Precision and recall satisfactory for lexically covered sentences
- No parses for out-of-vocabulary items owing to corpus size and specificity of preterminals, derived grammar not robust w.r.t. lexical coverage

<table>
<thead>
<tr>
<th>Anchor</th>
<th>Cov.</th>
<th>LR(tot.)</th>
<th>LP(tot.)</th>
<th>LR(cov.)</th>
<th>LP(cov.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEX</td>
<td>77.47</td>
<td>57.68</td>
<td>77.07</td>
<td>77.33</td>
<td>78.27</td>
</tr>
<tr>
<td>POS</td>
<td>98.12</td>
<td>76.42</td>
<td>78.36</td>
<td>77.92</td>
<td>78.44</td>
</tr>
</tbody>
</table>
Experiment 2

- 10-fold cross-validation over 3528 sentences from Verbmobil corpus
- Anchors of extracted trees (POS) only encode POS information
- Recall and precision satisfactory
- Valency and morphosyntactic information still encoded by way of tree derivation, including inflectional rules

<table>
<thead>
<tr>
<th>Anchor</th>
<th>Cov.</th>
<th>LR(tot.)</th>
<th>LP(tot.)</th>
<th>LR(cov.)</th>
<th>LP(cov.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEX</td>
<td>77.47</td>
<td>57.68</td>
<td>77.07</td>
<td>77.33</td>
<td>78.27</td>
</tr>
<tr>
<td>POS</td>
<td>98.12</td>
<td>76.42</td>
<td>78.36</td>
<td>77.92</td>
<td>78.44</td>
</tr>
</tbody>
</table>
Discussion

- Parseval measures achieved by derived LTIG comparable to performance of treebank-induced PCFG parsers:
  - Dubey & Keller, 2003 have trained a PCFG on subset of German NEGRA corpus, reporting 70.93% LP & 71.32% labelled recall (coverage: 95.9%)
  - Similar results obtained by Müller et al. (2003) on the same corpus (LP: 72.8%; LR: 71%)

- Current probabilistic parsing results for German in general less satisfactory than for English (cf. Dubey & Keller, 2003; Levy & Manning, 2003) differences most probably related to typological difference between languages
Summary

- First successful subgrammar extraction for German HPSG
- Method based on Chiang (2000) TAG extraction from Penn treebank
  - Definition of head-percolation and argument rules driven by HPSG principles, not heuristics
  - No treebank transformation necessary
- Performance of initial experiments promising: > 77% LP & LR
Future work

- Experiment with generalised/specialised node labels
- Multiply-anchored elementary trees
- Different parsing schemas

- Points to my current work
Using Dependency Treebanks as a source for extracting LTIGs

- There exists a number of dependency treebanks for different languages.
- They explicitly represent head/mod relationships.
- There is a natural relationship between dependency trees and derivation trees in TAG formalism.
- Might provide a tree decomposition operation for free.
- Try avoiding any language specific properties.
Starting point

- Dependency treebanks encoded in the so called CoNLL tree format.
- Transformation of CoNLL format into a PennTB like CF tree format.
Expression of the detoxication enzyme glutathione transferase P1-1 (GST P1-1) at elevated levels has been noted in many types of human tumors, including melanomas.
More formally: CoNLL trees

- A CoNLL dependency tree is a sequence $S$ of connected nodes $s_i$, $(1 \leq i \leq \text{len}(S))$ each of form:
  - $<M,H,\text{Dep}>$
    - "encoding the most relevant information"
    - where $M$ and $H$ are indices of elements $s_M, s_H \in S$
    - $\text{Dep}$ is the dependency relation between $s_M, s_H$
    - if $H < M$, we say that the head element is in left direction (denoted as LH); analogous for right head we use RH
  - $<0,\varepsilon,\varepsilon>$ for hidden root node
More formally: CF trees

- I call a target CF tree „linear dependency tree“ (LDT), and define it as a binary tree over a ranked alphabet $\Sigma$:
  - $x$, where $x \in \Sigma_0$ (terminal elements)
  - $x(t_1, t_2)$, where $x \in \Sigma_2$ (nonterminal elements)
  - $t_1$, $t_2$ are trees over $\Sigma$
- For the node labelling
  - $x \in \Sigma_2$ are further divided into disjoint sets
    - $X_{LH\_Dep}$, $X_{RH\_Dep}$
    - $x(t_1, t_2)$ into $x_{RH\_Dep}(t_{M'}, t_{H})$ and $x_{LH\_Dep}(t_{H}, t_{M'})$
The Transformation Algorithm

- **Core idea:**
  - Traverse a CoNLL sequence from left to right and construct a LDT incrementally bottom-up from the modifier elements to its heads.

- **Note:**
  - In general the head element of a modifier is not the adjacent right/left element, but might be a long-distant right/left element.

- **Because LDT is constructed bottom-up**
  - it might be that a tree must be adjoined into a larger tree.
Example
Ensuring proper spans

- It might happen that for a newly created nonterminal node the yield is not proper
  - if the right pos of node i, which stands left to another node j is greater than the left pos. of j
- Then:
  - create a new node with a trace element in order to ensure reversible mapping from LTD2CoNLL
  - copy and move corresponding subtrees
Extraction of LTIG from LTD

- Straightforward
  - cut of non-head subtrees
  - then define aux-trees as those which have a left/right yield node with same label as root

- Example LTIG-trees from Tiger TB:

  ((RH_CVC (:SUBST . LH_NK) (RH_PM (PTKZU "zu") (VVINF "bringen"))) 4 . 0.26666668)
  ((LH_NK (:RFOOT . LH_NK) (NN "Kurs")) 3 . 7.433102e-4)
Parsing: Efficient Early-style
LTIG parser

- Based on Schabes & Waters, 1995
- Extensions:
  - supports (disconnected) multi word lexical anchors
    - recursive trie traversal for lexical tree lookup
  - supports simultaneous adjunction at a single node
  - supports sharing nodes between trees
    - computes very compact forest of readings
  - two step unfolding of forest
    - extract all possible LTIG derivations (only anchors+tree indices)
    - expand indices to trees taking into account the LTIG operations that have been used
External format of LTIG grammars

(setq *start-symbols* '(s np))
(setq *ltig* '(
  ((s (:subst . np) (vp (v saw) (:subst . np))) 1 . 0.75)
  ((s (:subst . np) (vp (v saw))) (:subst . np)) 1 . 0.25)
  ((np (:subst . det) (n boy)) 1 . 0.5)
  ((det a) 1 . 0.5)
  ((n a) 1 . 0.5)
  ((np (:subst . det) (n woman)) 1 . 0.5)
  ((vp (v seems) (:lfoot . vp)) 1 . 0.5)
  ((vp (:rfoot . vp) (adv smoothly)) 1 . 0.5)
  ((vp (:rfoot . vp) (adv above) (:subst . np)) 1 . 0.5)
  ((vp (:rfoot . vp) (adv above)) 1 . 0.5)
  ((vp (XP (:rfoot . vp) (TO to)) (YP (adv slowly))) 1 . 0.5)
  ((n (adj nice) (:lfoot . n)) 1 . 0.25)
  ((n (adj tall) (:lfoot . n)) 1 . 0.5)
  ((n (adj pretty) (:lfoot . n)) 1 . 0.25)((vp (XP (:rfoot . vp)) (adv slowly)) 1 . 0.5)
))

Example trees from S&W, 95;
Same format for hand-crafted grammars & TB-based grammars;
When reading in, a lot of efficient indices are created;

Show Negra trees
Examples of parsing

- Extracting LTIG from first 1000 Tiger dependency trees
  - show LTIG grammar
  - do parsing
  - display trees

- Parsing time:
  - ~0.0372 sec/sentence computing & expanding all readings
  - ~17 words/sentence (ranging from 2 - 58)
Length of each sentence
Next steps

- **Transformation**
  - Check, whether for works for arbitrary non-projective cases (formally)

- **Experiments with as many languages as possible**

- **Parsing**
  - Improve tree filtering
  - Almost parsing ala Bangalore
  - Use of global statistical model ala Finkel et al. 2008 (they're using CRF)