Hybrid Parsing
ELSNET Summer School 2006

Kilian Foth, Wolfgang Menzel

Department for Informatics
Hamburg University

12. Oktober 2006

Overview

- Parsing
- Architectures
- Parsing as Constraint Satisfaction
- Weighted Constraint Dependency Grammar
- Information Fusion with Weighted Constraints

Parsing

assigning structural descriptions to sentences

Hybrid parsing
using a range of heterogeneous methods for parsing

Parsing

• Syntactic Structures
• Deep Parsing
• Shallow parsing

Syntactic Structures

most popular:

constituent structures

S

NP

D

N

V

P

PP

The quick brown fox jumps over the lazy dog

alternative view:

dependency structures

The quick brown fox jumps over the lazy dog

Why do we need syntactic structures?

→ guiding the semantic interpretation of an utterance

applications for ...

- information extraction
- machine translation
- corpus linguistics
- ...

different conventions for building trees and labelling them

→ annotation guidelines

treebanks: large collections of sentences annotated with trees

- English: Penn-Treebank
- Czech: Prague Dependency Treebank
- German: Negra-Treebank, Tiger-Treebank
- ...

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 1

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 2

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 3

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 4

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 5

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 6

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 7

Kilian Foth, Wolfgang Menzel

Hybrid Parsing: 8
Deep Parsing

- building recursively embedded tree structures
- building flat syntactic descriptions

**deep parsing**
- S → NP VP
- NP → D N 1
- N 1 → AP N 1
- AP → A
- N → N 1

**shallow parsing**
- S → NP VP
- NP → D N
- N → V NP
- V → (loves)
- N → Mary

((John) (loves (Mary)))

Deep Parsing

- realistic grammars are ambiguous:
  - N 1 → AP N 1
  - N 1 → N 1 NP
  - N 1 → N 1 PP
  - N 1 → N 1 A
  - N 1 → N PP

- lexical items are ambiguous:
  - V → jumps
  - N → jumps
  - A → brown
  - N → brown

- combinatorial search for a spanning tree licensed by the grammar

- worst case: ambiguities multiply out
  - extremely high degree of output ambiguity

Hinter dem Betrug werden die gleichen Täter vermutet, die während der vergangenen Tage in Griechenland gefälschte Banknoten in Umlauf brachten.

The perpetrators of this fraud are supposed to be the same as those who brought into circulation fake bills in Greece over the last few days.

- Paragram (Kuhn und Rohrer 1997): 92 readings
- Gepard (Langer 2001): 220 readings

Stochastic parsing

- trained on a treebank
- learning preferences from observations
- state-of-the-art systems are based on
  - markovization: generate rules by means of a stochastic model
  - lexikalization: condition probabilities on lexical items (e.g. phrase heads)
- pro: full disambiguation by determining the most likely structure
- but: loss of perspicuity, accountability and diagnostic capability

Paragram (Kuhn und Rohrer 1997):

![Graph 1](image1.png)

Gepard (Langer 2001):

![Graph 2](image2.png)
Shallow Parsing

- problem with deep parsing: the parser has to make decisions, without the grammar providing enough distinguishing information
- alternative: shallow parsing
  use simpler target structures to avoid decisions which cannot be taken reliably
- simpler model structures 
  mostly locally restricted relationships → machine learning techniques can be applied

Shallow Parsing

- different shallow components available: e.g.
- part-of-speech tagger
  The/DT quick/JJ brown/JJ fox/NN jumps/VB over/IN the/DT lazy/JJ dog/NN
- phrase chunker
  [The quick brown fox]NP [jumps]VP [over the lazy dog]PP

Overview

- Parsing
- Architectures
- Parsing as Constraint Satisfaction
- Weighted Constraint Dependency Grammar
- Information Fusion with Weighted Constraints

Architectures

- tight integration
  combining models in a single (global) decision space
  → no local decisions in the preceding component
  e.g. word recognition (Jelinek 1976)

- loose integration
Tight integration

- e.g. stochastic machine translation (Brown et al. 1990)

```
language model
lexical transfer
fertility model
```

- e.g. dependency parsing (Wang and Harper 2002)

```
parser
supertagger
```

Loose integration

- suboptimal decisions are avoided
- but: tight integration cannot always be easily achieved
  - models must be of the same type (i.e. probabilistic)
  - simple mapping between the structures dealt with by the different models
  - search spaces must be (efficiently) combined

```
POS tagger
parser
```

possible remedies:
- learn to correct errors
  The subsequent component is trained on the erroneous output of the preceding one.
- select from a number of alternatives (pipeline)
  e.g. multitagging

```
POS multi tagger
parser
```

Examples of Hybrid Parsers

- XTAG (Srinivas 1995):
  - filtering and guiding via POS tags and supertags
  - consistency checks find provably impossible eltrees
  - only the top 3 elementary trees are used for parsing...
  - ... unless parsing fails altogether
  - global preferences e.g. for low PP attachment

- Heart of Gold (Callmeier et al. 2004):
  - XML-based middleware integrates the PET HPSG parser, named entity detector, shallow clause detector, shallow parser SPPC, lexical semantics (GermaNet), stochastic topology parser, and POS tagger
  - no fixed order of processing (blackboard)
  - shallow components used for their robustness and throughput
Examples of Hybrid Parsers

- **Dienes & Dubey 2003**: find and resolve empty elements in phrase structure trees
  - a maximum-entropy trace tagger finds trace locations
  - a PCFG attaches dislocated elements in those places
  - better than entrusting both tasks to the PCFG

- **Charniak & Johnson 2005**: feature-rich PCFG parsing
  - a simplified PCFG generates a packed parse forest of likely candidates
  - the forest is pruned by marginal properties
  - the full PCFG is ranks the remaining possible trees
  - new record for parsing the WSJ corpus

- **Zeman & Zabokrtsky 2005**: dependency parsing of Czech
  - seven different parsers are run in parallel (ensemble)
  - individually, 64% to 85% accuracy
  - various combination policies are investigated
  - weighted voting: trust each prediction in proportion to the overall reliability of its source
  - altogether, 87% structural accuracy
  - “Diversity of opinion is more important to success than individual excellence.”

Overview

- Parsing
- Architectures
- Parsing as Constraint Satisfaction
- Weighted Constraint Dependency Grammar
- Information Fusion with Weighted Constraints

Constraint Satisfaction

- A constraint is a piece of declarative knowledge which restricts the solution space of a given problem.

- How is the structure of the solution space defined?
  - How many variables?
  - Is this number known in advance?
  - Are variables introduced dynamically?
  - Which kind of values can be attached to variables?

The CS View

- **Tsang 1993**
  - A fixed number of variables is bound to different data types
  - Variables have a predefined meaning
  - Different variables can have different domains
  - Constraints are n-ary relations between specific variables, i.e. subsets of the corresponding Cartesian product of their domains

A popular Constraint Satisfaction Problem

Each row, column and subsquare must contain all 9 digits.
- 81 variables, 9 values, 30 unary / 27 9-ary constraints

The HPSG View

- **Pollard and Sag 1987**
  - Only a single variable with a recursive feature structure as its value
  - Feature structures may again contain variables (e.g. to establish coreference).
  - Constraints are implications over feature structures
  - Require to unify a feature structure with the consequence if it is subsumed by the premise
  - Embedded variables are instantiated (information accumulation)
The HPSG View

- structure construction as part of information accumulation
- alternative interpretations (variable bindings)
  - use underspecification
  - need to be enumerated

The LP View

- constraint handling rules (Frühwirth 1992)
- application to NLP: CHR-Grammar (Christiansen 2002)
- constraints
  - are derived from the input and the grammar
  - describe the structure of the input
  - high-level expectations can be effectively integrated
  - difficulties with alternative interpretations of an input sentence

Other Views

- Property Grammar (Blache 1996)
  - elements from a fixed set of constraints describe NL input in a bottom-up manner
  - constructions are identified based on constraints holding for a candidate set of words
  - problem: where do the candidate sets come from?
- Mozart (Oz) (Duchier 2001)
  - grammar is described by means of set-valued constraints (dependency structures)
  - multi-level representations (Debussman and Duchier 2004)
    - syntax, topological fields, predicate-argument structure, scopus

Constraint Grammar

- typical CS problem:
  - constraints: conditions on the (mutual) compatibility of dependency labels
  - indirect definition of well-formedness: everything which does not violate constraint explicitly is acceptable
  - strong similarity to tagging procedures

- two important prerequisites for robust behaviour
  - inherent fail-soft property: the last remaining category is never removed even if it violates a constraint
  - possible structures and well-formedness conditions are fully decoupled: missing grammar rules do not lead to parse failures
  - complete disambiguation cannot always be achieved

- size of the grammar (English): 2000 Constraints
- quality:

<table>
<thead>
<tr>
<th></th>
<th>without heuristics</th>
<th>with heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>95.5%</td>
<td>97.4%</td>
</tr>
<tr>
<td>recall</td>
<td>99.7...99.9%</td>
<td>99.6...99.9%</td>
</tr>
</tbody>
</table>

- Tapanainen & Järvinen 1995:
  - reuse Karlsson’s Constraint Grammar unchanged
  - add handwritten rules that generate specific dependencies
  - report 95.3%/87.9% dependency accuracy for Bank of English text

Constraint Dependency Grammar

- Maruyama 1990
  - each word form of a sentence corresponds to a variable.
  - number of variables is a priori unknown.
  - no predefined meaning for variables.
  - every constraint must hold for each variable or a combination thereof.
  - all value assignments for all variables are taken from the same domain: $W \times L$ (attachment values).
  - fully specified dependency relations $D \in W \times W \times L$
  - originally invented to express all possible preposition attachments concisely for Japanese
Constraint Dependency Grammar

- lexicon items and levels of analysis define the conceivable structures
- constraints make linguistically motivated restrictions
- an assignment which satisfies all constraints is by definition a solution
- → parsing is structural disambiguation

complete Constraint Satisfaction procedure

- removal of incompatible dependency edges
- constraint propagation via Waltz-Filtering

interactive disambiguation: Increasingly domain specific constraints are applied if no full disambiguation can be achieved (Maruyama 1990)

CDG is mildly context sensitive

time complexity: \( \mathcal{O}(|C| \cdot n^4) \)

- \( n \) length of the input
- \( C \) constraint set

Extraction of all parses can be NP complete!

Hypothesis Space

<table>
<thead>
<tr>
<th>root</th>
<th>root</th>
<th>root</th>
<th>root</th>
<th>root</th>
</tr>
</thead>
<tbody>
<tr>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>det</th>
<th>det</th>
<th>det</th>
<th>det</th>
<th>det</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>subj</th>
<th>subj</th>
<th>subj</th>
<th>subj</th>
<th>subj</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>obj</th>
<th>obj</th>
<th>obj</th>
<th>obj</th>
<th>obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>DetNom</th>
<th>Det</th>
<th>Verb</th>
<th>Subj</th>
<th>Obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>DetNom</td>
<td>Det</td>
<td>Verb</td>
<td>Subj</td>
<td>Obj</td>
</tr>
</tbody>
</table>

\( \text{Der Mann besichtigt den Marktplatz} \)
Benefits of Constraints for Parsing

- ideally suited for information fusion
  - multiple information sources are crucial to NLU
  - all sources can be integrated into the same computation
  - but each rule need only deal with one source (modularity)
- potential for fail-soft behaviour
  - rules can denote preferences as well as laws
  - preferences of different strengths can be modelled
- no equivalence of structures and rules necessary
  - for English, all allowed configurations can be listed
  - in other languages it is easier to describe the forbidden configurations instead
  - allows detailed diagnoses such as “syntactically correct, but misinflected”
Overview

- Parsing
- Architectures
- Parsing as Constraint Satisfaction
- Weighted Constraint Dependency Grammar
- Information Fusion with Weighted Constraints

Local Constraints

A rule from DIAGRAM (ROBINSON 1982):

```
CONSTRUCTOR (NOMHD (COND (\(D\)) \(NBR\) \(@NBR\) \(NCOMP\) \(@NCOMP\)) \(mass\) \(w\) \(\!\!\!\!\!\!\!\!\!
```

Nonlocal Constraints

Constraints can also capture information from different levels of analysis:

- Quantor plausibility: if the VP is modified by ‘usually’, then the quantor ‘every’ is implausible
- Euphony: two successive words should not have the same phonetic form
- Style: the word forms in a sentence usually belong to the same register and epoch

Weighted Constraints

Why weighted constraints?

- Weights help to fully disambiguate a structure.
  - Hard constraints are not sufficient (HARPER ET. AL. 1995).
- Many language regularities are preferential and contradictory.
  - extraposition
  - linear ordering in the German mittelfeld
  - topicalization
- Weights are useful to guide the parser towards promising hypotheses.
- Weights can be used to trade speed against quality.

Extensions

- relational view on dependency structures instead of a functional one:
  - SCHÖRDER (1996): access to lexical information at the modifying and the dominating node
- recognition uncertainty / lexical ambiguity
  - HARPER AND HELZERMAN (1996): hypothesis lattice additional global constraint (path criterion) introduced
- existence quantors and long-distance dependencies
  - FOTH (2002): arbitrary global constraints more than two edges can be restricted in some configurations

Weighted Constraints

- penalty factors reduce the preference for hypotheses which violate a constraint
- w(c) = 0: hard constraint, must always be satisfied e.g. licensing structural descriptions
- 0 < w(c) < 1: soft constraint may be violated if no better alternative is available
  - w(c) << 1: strong, but defeasible well-formedness conditions
  - w(c) >> 0: defaults, preferences, etc.
- w(c) = 1: no effect, neutralizes the constraint
- penalties can also depend on the specific subordination, e.g. the closer, the better

Weights In Collision

A phrase from Roman poetry: "… nympham amabat sol …"
Examples of Constraint Weights

Some reasonable assumptions about English:

- Subjects and objects appear only under verbs; \( w(c) = 0 \)
  "We won." / "We winners."
- Finite verbs almost always have subjects; \( w(c) \approx 0 \)
  "And so it goes." / "And so goes."
- Infinitives should not be split; \( 0 < w(c) < < 1 \)
  "Try not to think of it." / "Try to not think about it."
- Transitive verbs usually have objects; \( 0 < < w(c) < 1 \)
  "We sell cars at fair prices." / "We sell at fair prices."
- Plural nouns are slightly rarer than singulars; \( w(c) \approx 1 \)
  "I feed the fish/sg." / "I feed the fish/pl."

Accumulating Scores

- accumulating (multiplying) the weights for all constraints violated by a partial structure
  → both single dependency relations and tuples have combined scores
- local scores are multiplied into a global one
  \[
  w(t) = \prod_{c \in \text{constraint}(t,c)} w(c) \cdot \prod_{(e_i,e_j) \in \text{tuple}(e_i,e_j),c} w(c)
  \]
- determining the optimal global structure
  \[
  t(s) = \arg \max_w w(t)
  \]
  → parsing becomes a constraint optimization problem

Comparison to Optimality Theory

- There is no generative backbone.
  → Everything that is not forbidden is possible.
- Scores are accumulated.
  → Several weak constraints can gang up on a stronger one.
- Rules, not principles, are scored.

Constraint Optimization

- consistency: works only for hard constraints
- pruning: successively remove the least preferred dependency relations
- search: determine the optimum dependency structure
- structural transformation: apply local repairs to improve the overall score

Note: According to the \( \Lambda \)^P hypothesis, there can be no efficient solution method for WCDG!
Search

Structural Transformation

- elementary repair operations:
  - change a subordination
  - change the label of an edge
  - choose a lexical reading
- many degrees of freedom:
  - how many changes during one step?
  - how many alternative steps are tried at a time?
  - which transformation is tried first?
  - is the selection (partially) random?
  - when do we give up?
Structural Transformation

Usually local transformations result in unacceptable structures
- sequences of repair steps have to be considered.
- e.g. swapping SUBJ and DOBJ

<table>
<thead>
<tr>
<th>a)</th>
<th>syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>det/2</td>
<td>der</td>
</tr>
<tr>
<td>man/3</td>
<td>besichtigt</td>
</tr>
<tr>
<td>den/5</td>
<td>marktplatz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b)</th>
<th>syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>det/2</td>
<td>der</td>
</tr>
<tr>
<td>man/3</td>
<td>besichtigt</td>
</tr>
<tr>
<td>den/5</td>
<td>marktplatz</td>
</tr>
</tbody>
</table>

Frobbing
- gradient descent search
- escaping local minima: increasingly complex transformations → local search
- heuristically guided tabu search
  - transformation with perfect memory
  - propagation of limits for the score of partial solutions
  - faster than best-first search for large problems
  - inherently anytime
  - to date the best option for solving WCDG

*Frobbing: randomly adjusting the settings of an object, such as the dials on a piece of equipment or the options in a software program. (The Word Spy)
Guided Local Search

- memory as energy landscape defined over the hypothesis space (Vodouris 1997)
- transformation with imperfect memory (Schulz 2000)
- augmented scoring function:

\[ h(s) = g(s) + \lambda \cdot \sum_{i=1}^{w} n_i \cdot I_i(s) \]

- utility: where to change the weights of the scoring function

\[ \text{util}(s, f) = I' \cdot C_i \frac{1}{1 + n_i} \]

- policy:
  - high costs → high utility
  - repeated repair → lower utility

Overview of Solution Methods

- Consistency: makes no mistakes, but leaves far too many choices (bad precision)
- Pruning: leaves fewer choices, but can destroy the solution (bad recall)
- Search: correct and complete, but unaffordable (bad time and space complexity)
- Heuristic Transformation: incomplete, but good in practice, and interruptible (bad theoretical foundation)
- Chart-like bottom-up parsing: avoids some inefficiency of backtracking, but not enough (good idea, bad in practice)
- Genetic algorithms: semi-randomized transformation, takes far too long to converge
- Shift-reduce parsing: more of that later

Modelling Syntax with Constraints

- WCDG ideally supports grammar development by providing diagnostic information
- overgeneration appears as high scores for wrong analyses
- undergeneration appears as low scores for the right analysis
- the responsible constraints are immediately obvious
- typical development cycle:
  1. parse a sentence with a draft grammar
  2. correct the structure manually (and store it)
  3. change the constraints violating the gold standard
  4. introduce constraints prohibiting misanalyses
  5. parse the sentence with the modified grammar
  6. repeat

Examples of WCDG Constraints

// Subjects and objects appear only under verbs
\( X \text{[Syntx]} : \text{‘subject definition’} : 0.0 : \)
\( X \text{[label = subject -> X}} \text{[cat = VB;}} \text{]) \)

// Finite verbs almost always have subjects
\( X \text{[Syntx]} : \text{‘missing subject’} : 0.1 : \)
\( X \text{[cat = VBP -> has(X[id, subject]);}} \text{]) \)

// Infinitives should not be split
\( X \text{[Syntx/VT/Syntx]} : \text{‘split infinitive’} : 0.2 : \)
\( X \text{[word = to & X[cat = VBP -> Y[from < X[from;}} \text{])} \)

// Transitive verbs usually have objects
\( X \text{[Syntx]} : \text{‘missing object’} : 0.8 : \)
\( X \text{[cat = VBP & exists(X[transitive -> has(X[id, dobject]);}} \text{])} \)

// Plural nouns are slightly rarer than singulars
\( X \text{[Syntx]} : \text{plural} : 0.99 : \)
\( X \text{[number != plural;}} \text{]) \)

The Grammar of German

- only two levels: syntax, reference
- about 1000 handwritten constraints
- allows non-projective dependency structures if necessary
- strongly lexicalized:
  - e.g. valence information for verbs and prepositions

An overview of relation types: ADV, APP, ATTR, AUX, AVZ, CJ, DET, ETH, EXPL, GMOD, GRAD, KOM, KON, KONJ, NEB, NP2, OBJA, OBJA2, OBJD, OBJG, OBJC, OBJI, OBJP, PAR, PART, PN, PP, PRED, REL, S, SUBJ, SUBJC, VOK, ZEIT, ”.

"They made us an offer we could not refuse."
some ambiguity is meaningful, and some is spurious

spurious ambiguity should be normalized
(if you’re concerned about parsing accuracy):
\{X\!SYN/\!SYN\} : 'VP lowering' : 0.1:
X.label = AUX -> Y.label != 'ADV;

is this a useful syntax annotation?

Projectivity constraints are still needed for other phenomena
→ make exceptions for coordinations
but know where to stop:

Subjects are nouns:
→ a single subject is allowed:

Unfortunately, there are many exceptions:

- Hätte ich ihr vertraut, ich hätte sie nicht verloren.
- Was es auch ist, das Phänomen muß untersucht werden.
- Wenn das Salz nicht mehr salzt, womit soll man salzen?
- Daß es so ein Erfolg würde, damit rechnete niemand.
- Scheint es gleich wir, so hat es doch Methode.
- Freilich, der Bundesrat muß noch zustimmen.
- Vom Abend bis zum Morgen geht die Feier.
- Das jedoch scheint fraglich.
- Wir im- und exportieren Hardware.

When two principles clash, you can simply write two constraints with different weights
where applicable, the stronger constraint will ‘win’
but it is better to allow special cases explicitly
one rule cannot make exceptions from another rule
→ all exceptions must be part of the rule itself
the real ‘Vorfeld’ constraint is 86 lines long!
many rules consist predominantly of exceptions
→ generativity via back door

WCDG allows logical dependency to be expressed as physical dependency:

just don’t enforce projectivity

Many conditions concern more than two dependencies
but high-arity constraints are extremely expensive
→ approximate the condition with several binary constraints
introduce additional levels (MARUVAMA)
introduce operators that extend binaryness in a controlled way
Example: German vorfeld
“If two constituents precede the finite verb, then the verb itself is not labelled as S (main clause).”
→ this involves three dependency edges!
→ but the third is always the parent of the two others

Subjects are nouns:
→ only a single subject is allowed:

Sample Constraints

Subjects are nouns:
→ only a single subject is allowed:

Examples:

- ‘Das ist gut.’
- ‘Demnächst ist gut.’
- ‘Demnächst ist gut!’

X\!SYN : 'SUBJ-Kategorie' : category : 0.0 :
X.label = SUBJ ->
\{isa(X, Nominal) & X.cat != 'PRF | X.cat = 'ADJA | quoted(X);\}

Only a single subject is allowed:
→ ‘Ich hatte viel Bekümmeris.’

X\!SYN/\!SYN : 'doppeltes Subjekt' : uniq : 0.0 :
\{subsumes(Label, Subjekt, X.label)
  -> 'subsumes(Subjekt, Subjekt, Y.label);\}
Sample Constraints

The subject is least oblique argument:

```plaintext
// 'Heute tanzt der König das Menuett.'
// '*Heute tanzt das Menuett der König.'
{X/SYR}/Y/SYR) : 'Subjekt-Position' : order : 0.9 :
X.label = SUBJ & subsurnes(Label, Nominalobjekt, Y.label)
->
X/from < Y/from |

// 'falle sich/OBJA nicht ein Investor/SUBJ findet'
Y/cat = PRF |

// 'ein Mann, den/OBJD man/SUBJ vertrauen kann'
Y/cat = PAS | Y/cat = PRELS | Y/cat = PWAT |
Y/cat = PWAV | Y/case = PRELAT |
has(Y.id, find_initial));
```

The Lexicon

- full-forms for all closed-class items
- 8,500 verb stems, 27,000 noun stems
- compound analysis
- lexical templates for unknown words

```plaintext
dar := [cat:AKT, case:nom, number:sg, gender:masc, definite:yes ];
ganz := [cat:ADV,subcat:grade, likes_positive:yes,
 modifies: <Adjective,1,Adverb,1, Prozession,1,KOKOM,1,Verb,0.99> ];
ausgelosten := [cat:ADJA,base:ausholen,partizipial2:yes,
 degree:positive,avz:allowed,perfect:haben,
 valence:'a?',case:gen_dat,
 number:sg,gender:bot,flexion:weak,suffix:en];
```

```
'^[A-Z]\$' =~
{X!SYN} : 'Subjekt-Numerus' : agree : 0.1 :

// 'Wir gehe in den Zoo.'
// 'Das sind ganz übel Gesellen!'
// 'Was für ein gutes Jahr.'
// '80 Prozent mehr wurden verkauft als im letzten Jahr.'
```
Relative Importance of Information Sources

<table>
<thead>
<tr>
<th>Class</th>
<th>Purpose</th>
<th>Example</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>init constraints</td>
<td>appositions are nominals</td>
<td>3.14</td>
</tr>
<tr>
<td>pos</td>
<td>POS tagger integration</td>
<td>prefer the predicted category</td>
<td>1.77</td>
</tr>
<tr>
<td>root</td>
<td>root subordinations</td>
<td>only verbs should be tree roots</td>
<td>1.72</td>
</tr>
<tr>
<td>cat</td>
<td>category cooccurrence</td>
<td>adverbs do not modify each other</td>
<td>1.13</td>
</tr>
<tr>
<td>order</td>
<td>word-order</td>
<td>determiners precede their regents</td>
<td>1.11</td>
</tr>
<tr>
<td>proj</td>
<td>projectivity</td>
<td>disprefer nonprojective coordination</td>
<td>1.09</td>
</tr>
<tr>
<td>exist</td>
<td>valency</td>
<td>finite verbs must have subjects</td>
<td>1.04</td>
</tr>
<tr>
<td>punct</td>
<td>punctuation</td>
<td>subclauses are marked with commas</td>
<td>1.03</td>
</tr>
<tr>
<td>agree</td>
<td>rection and agreement</td>
<td>subjects have nominative case</td>
<td>1.02</td>
</tr>
<tr>
<td>lexical</td>
<td>word-specific rules</td>
<td>&quot;entweder&quot; requires &quot;oder&quot;</td>
<td>1.02</td>
</tr>
<tr>
<td>dist</td>
<td>locality principles</td>
<td>prefer short attachments</td>
<td>1.01</td>
</tr>
<tr>
<td>pref</td>
<td>default assumptions</td>
<td>assume nominative case by default</td>
<td>1.00</td>
</tr>
<tr>
<td>sort</td>
<td>sortal restrictions</td>
<td>&quot;win&quot; takes only local predicatives</td>
<td>1.00</td>
</tr>
<tr>
<td>uniq</td>
<td>label cooccurrence</td>
<td>there can be only one determiner</td>
<td>1.00</td>
</tr>
<tr>
<td>zone</td>
<td>crossing of marker words</td>
<td>conjunctions must be leftmost dependents</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Availability

The reference implementation of WCDG is Free Software.

- online demo
  [http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo](http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo)
  - Does only time-limited analysis (for interactivity)
  - Contact us for bulk parsing

- download
  [http://nats-www.informatik.uni-hamburg.de/download](http://nats-www.informatik.uni-hamburg.de/download)
  - Runs on x86-Linux; ports are planned
  - Contains program, grammar and annotation manual of German

Stochastic Helper Components

- Why are handwritten rules not enough?
  - Language understanding is largely guided by preferences
  - In particular, preferences between alternatives that are both "correct"
  - Intuitive knowledge is not easily made explicit
  - Empirical models can capture it more reliably

- Trying to gain the best of both worlds
  - Helpers: POS Tagging, Supertagging, Chunk parsing, PP attachment, Shallow dependency parsing

Hybrid Parsing: POS Tagger

- The grammar rules disambiguate most sentences correctly…
- …assuming we know the word categories
- Wide-coverage parsing requires an extremely broad view of categories:
  - "Die Xeon-Prozessoren mit 256 KByte L2-Cache auf dem Die brauchen 133 MHz Front-Side-Bustakt."
- Even closed-class items like "die" might be something different!

- POS tagging is well-understood
- It’s not perfect, but we don’t have believe it completely

Method:
- Call TnT ([Brants 1996](http://nats-www.informatik.uni-hamburg.de/Papa/ParserDemo)) on a sentence before parsing
- Map its probabilities to scores
- Prefer the predicted categories:
  - \{X:SYN\} : tagger : [ predict(X|id, POS, X|cat) ] : predict(X|id, POS, X|cat) = 1.0;
- Effect: smaller hypothesis space, better guidance towards probable solutions
- Typically more than halves the error rate
- POS tagging is an enabler for large-scale WCDG
Hybrid Parsing: POS Tagger

Intricacies of tagger-parser integration:

- TnT and WCDD use different lexica → suppress out-of-lexicon predictions
- TnT calculates probabilities, WCDD uses penalties → normalize the highest $p$ to 1
- TnT can use different beam widths → find a suitable one through experimentation
- Tagger errors can propagate to the parser → this one needs more effort

Hybrid Parsing: Tagger Errors

- TnT makes around 5% errors on unseen input
- 2% of these are hard errors
- Some of them can be overridden . . .
- . . . but many can’t:

One tagger error causes three attachment errors

Hybrid Parsing: Hybrid Tagging

- Many tagger errors are obvious to a human expert: “Die/ART Organisation/NN hatte/VAFIN am/APPRART Dienstag/NN einen/ART Waffenstillstand/NN erklärt/VVFIN.” There are virtually never two finite verbs in a clause!
- By combining lexicon and sentence-global knowledge, we can figure out the truth: ‘erklärt’ is a past participle!
- But neither trigrams nor constraints can easily express this
- The answer is (of course) hybrid pre-processing.

Hybrid Parsing: Hybrid Tagging

- Apply automatic correction rules to TnT’s output:
  - If two finite verbs co-occur, one of them is really infinite
  - “als” is not a conjunction if the verb is in the middle of the clause
  - An oblique personal pronoun near the corresponding base form is almost certainly a reflexive pronoun instead
  - Words in CamelCase are almost certainly proper nouns
- Over 50 rules are used altogether
- Tagging accuracy rises from 97.2% to 97.7% (on NEGRA)
- Parsing accuracy rises from 89.0% to 89.7% (of a possible 90.4%)

Hybrid Parsing: Supertagging

- Supertagging (JOSHI 1999) extends tagging
- also predicts relation type, attachment direction, child nodes
- invented for LTAG, which is quite similar to WCDD
- used as a filter, it proved an enabling technology there
- But WCDD does not use elementary trees, only edges
- How to adapt subtree prediction to WCDD?

Hybrid Parsing: Supertagging

- A supertag might predict
  - the edge label
  - the subordination direction (left/right/not)
  - labels of complements (pre- and postmodifiers)
  - labels of all modifiers
  - the exact sequence of modifiers
- We define four subpredictions that can be made: label, direction, premodifiers, postmodifiers
- different combinations of these can be tested
- accuracy can be measured by exact tag or by subprediction

Hybrid Parsing: Supertagging

- transform the NEGRA corpus into dependency format
- extract a generalized supertag for each word: PP+S/N+AUX,KON,SUBJ
- project these onto the various features sets
- re-train TnT on these data
- call this model before parsing
- integrate the predictions with four new constraints, e.g.:

$$\{X:SYN\} : 'ST:direction' : stat : 0.9 : X/ & predict(X↓id, ST, dir) = R | X\ & predict(X↓id, ST, dir) = L | X| & predict(X↓id, ST, dir) = N;$$
Hybrid Parsing: Supertagging

- The best supertag model J increases structural parsing accuracy from 89.3% to 91.9%
- surprising: big improvement even though 1/3 of all supertags are wrong
- of course, perfect supertags would bring us to 97.2%
- \(\rightarrow\) supertagging is rightly called "almost parsing"
- Nevertheless, we can now directly benefit from future supertag research

Hybrid Parsing: Chunk Parsing

- Abney 1991: two-stage parsing model "[When I read] [a sentence],
  [I read it] [a chunk] [at a time]."
- syntax within chunks is regular, chunk attachment is more complex
- advantage: small-scale ambiguities are not multiply combinatorially
- has been successfully used to speed up some parsers
- WCDG could profit e.g. from noun phrase detection

Hybrid Parsing: PP Attachment

- almost one word in 30 is a mis-attached preposition
- the problem is well-known to be difficult
- many different factors contribute to PP attachment
- some of them are very hard to formalize
- (all the following examples also work in German)
- \(\rightarrow\) "through" cannot modify "girl" because of projectivity
- \(\rightarrow\) "the statue in the harbour was made in France."
- \(\rightarrow\) "in" cannot modify "was", because "statue" already does

Spot the obvious weak link in our rule set!
Hybrid Parsing: PP Attachment

- "The bill was vetoed by the House of Lords."
  → "House of Lords" is a fixed expression
- "This chair was instituted on the orders of the king."
  → "on the orders" is incomplete without a preposition
- "The holding bought 1,000,000 shares for $15 a share."
  → "shares for $15 a share" would be bad style
- "Please wash the infection with soap."
  → "infection with soap" would make no sense
- "I bought a stamp for sixpence."
  → "for" might modify either "bought or "stamp" with no meaning change

Hybrid Parsing: PP Attachment

- So far we use only syntactic and some idiom rules
- Many of the other criteria are almost inexpressible
- Some of them might be approximated by simple lexicalization
- But word-specific constraints would number many millions
- Again, empirical knowledge might be helpful

Hybrid Parsing: Shift-Reduce Parsing

Here is a sentence that appears to be not particular difficult:

"The Commission compiles yearly reports about the state of the realization of the goals named in III-209, and about the demographic situation in the Union."

Hybrid Parsing: Shift-Reduce Parsing

Yet it is analysed quite wrongly by our parser.

(Well, not really — only if you give it too little time.)
even a much simpler model could have gotten this sentence right
but simple models don’t have all the nice coverage
again, we should combine advantages of both worlds
a fast parser could deliver an initial guess . . .
. . . and transformation can choose a better solution if it finds one
this would improve both anytime behaviour and accuracy in the limit

we choose a deliberately simple model
shift-reduce parsing (Nivre 2003) creates projective dependency trees in linear time
idea: at each word, on of four moves can be made: shift onto stack, reduce from stack, attach to the right, or attach to the left

we must train a parse move predictor for a given state
by iterating it, we get a predictor for complete trees
optionally, we could annotate LEFT and RIGHT with edge labels
what features in a parse state could we use for training?
the top stack word (word form or POS tag)
the context of this word (its regent and its dependents so far)
the next input word
the distance between both words
the words in a fixed lookahead window

again, constraints prefer edges that match the predictions:
\( \{X!SYN\} : \text{SR:regent} : \text{stat} : 0.9 : \text{predict}(X↓id, \text{SR}, \text{gov}) = X↑to; \)
\( \{X!SYN\} : \text{SR:NIL} : \text{stat} : 0.9 : \text{predict}(X↓id, \text{SR}, \text{gov}) = 0; \)
\( \{X!SYN\} : \text{SR:Label} : \text{stat} : 0.9 : \text{predict}(X↓id, \text{SR}, \text{lab}) = X.label; \)
the 0.9 was tuned exhaustively
it guarantees that transformation usually starts from the exact oracle parse

parsing accuracy improves from 89.3% to 91.5%
both parsers in combination are a lot better than either
WCDG runs an order of magnitude faster at the same quality level
makes you wonder what a really good oracle parser would do for us
unfortunately, the published German stochastic parsers all generate phrase-structure
what could be improved about our oracle?

we choose a deliberately simple model
shift-reduce parsing (Nivre 2003) creates projective dependency trees in linear time
idea: at each word, on of four moves can be made: shift onto stack, reduce from stack, attach to the right, or attach to the left

Here are the correct sequence of moves:

Here are some obvious defects:
How can it be a GMOD without a genitive determiner?
Again, our model does not look that far into the context
This could be captured with a corner feature (Yamada et al. 2003)

Why is the subclause assumed to be a fragment?
Well, that’s what we have to do to our treebank before training...
Newer approaches allow nonprojective training data (Nivre 2005)

What if we use more than one predictor?
Accuracy

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Predictors</th>
<th>structural</th>
<th>labelled</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>none</td>
<td>72.0%</td>
<td>58.3%</td>
</tr>
<tr>
<td>2</td>
<td>TnT</td>
<td>89.0%</td>
<td>87.1%</td>
</tr>
<tr>
<td>3</td>
<td>hybrid POS</td>
<td>89.3%</td>
<td>87.5%</td>
</tr>
<tr>
<td>4</td>
<td>POS+CP</td>
<td>89.8%</td>
<td>88.0%</td>
</tr>
<tr>
<td>5</td>
<td>POS+ST</td>
<td>91.9%</td>
<td>90.5%</td>
</tr>
<tr>
<td>6</td>
<td>POS+PP</td>
<td>90.6%</td>
<td>89.9%</td>
</tr>
<tr>
<td>7</td>
<td>POS+SR</td>
<td>91.5%</td>
<td>89.8%</td>
</tr>
<tr>
<td>8</td>
<td>POS+PP+SR</td>
<td>91.4%</td>
<td>89.6%</td>
</tr>
<tr>
<td>9</td>
<td>POS+PP+ST</td>
<td>92.0%</td>
<td>90.6%</td>
</tr>
<tr>
<td>10</td>
<td>POS+ST+SR</td>
<td>92.2%</td>
<td>90.7%</td>
</tr>
<tr>
<td>11</td>
<td>all five</td>
<td>92.3%</td>
<td>90.9%</td>
</tr>
</tbody>
</table>

three predictors are even better than two
all five predictors are best
spot the exception!

POS tagging is an enabling technology for WCDG
stochastic models can replace grammar writing with data collecting
cheap, simple empirical models can usefully complement a heavy-weight deep model
two parsers are better than either one
good heuristics can massively affect the time/space trade-off


Syntactic category prediction is very useful, but morphology predictions would be even more useful
German adjectives have up to 26 underlying feature combinations
In simulation, morphology information adds another 1% of parsing accuracy
However, that is with an error-free morphology tagger
How good must a morphology component be to yield a net benefit?
Future Work: Nuclei

- Nuclei are actually an ancient concept in dependency grammar (Tesnière 1959)
- could be useful for other things than proper names:
  - verb phrases: “Ich weiß nicht, was ich [hätte tun sollen].”
  - category-changing idioms: “Es sieht [alles andere als] gut aus.”
- a preprocessor could replace known nuclei with new word hypotheses
- problem: idioms can have compositional homonyms
- solution: WCDG can already deal with alternatives in lattices

Future Work: Solution Methods

- Feature-based stochastic dependency tree learning (MacDonald et al. 2005)
  - Edge probabilities are learnt solely through unary features.
  - ... but over 13,000,000 of them
  - Extra work to guarantee the result is a tree
- Example-based parsing (Kong et al. 1998)
  - We have many thousands of ready-parsed sentences
  - Yet we often fail to produce good results for very similar sentences if they are very long
  - The known structure should be utilized at least as an initial guess

Future Work: Applications

- extend the reference resolution capabilities
  - So far we do only relative pronouns
  - But personal pronouns, possessives, nouns and even verbs can also refer
  - Problem: antecedents are often found in previous sentences
- test psycholinguistic adequacy claims on the parser
  - needed: left-to-right incrementality
  - needed again: multi-level representations
- use the diagnostic ability for language learning purposes
  - optimizing the grammar for non-native language
  - disambiguating multi-level representations

Future Work: Nuclei

- Nuclei are actually an ancient concept in dependency grammar (Tesnière 1959)
- could be useful for other things than proper names:
  - verb phrases: “Ich weiß nicht, was ich [hätte tun sollen].”
  - category-changing idioms: “Es sieht [alles andere als] gut aus.”
- a preprocessor could replace known nuclei with new word hypotheses
- problem: idioms can have compositional homonyms
- solution: WCDG can already deal with alternatives in lattices

Future Work: Solution Methods

- Feature-based stochastic dependency tree learning (MacDonald et al. 2005)
  - Edge probabilities are learnt solely through unary features.
  - ... but over 13,000,000 of them
  - Extra work to guarantee the result is a tree
- Example-based parsing (Kong et al. 1998)
  - We have many thousands of ready-parsed sentences
  - Yet we often fail to produce good results for very similar sentences if they are very long
  - The known structure should be utilized at least as an initial guess

Future Work: Applications

- extend the reference resolution capabilities
  - So far we do only relative pronouns
  - But personal pronouns, possessives, nouns and even verbs can also refer
  - Problem: antecedents are often found in previous sentences
- test psycholinguistic adequacy claims on the parser
  - needed: left-to-right incrementality
  - needed again: multi-level representations
- use the diagnostic ability for language learning purposes
  - optimizing the grammar for non-native language
  - disambiguating multi-level representations