A walk in the forest

Johan Benum Evensberget

January 18, 2012

Eagle's eye perspective

- HPSG and broad-coverage precision grammars can provide a high quality analysis of natural language
- Processing times are comparatively quite slow
- Unsuitable for large amounts of data on commodity hardware
- Unsuitable when fast analysis time is required

Can context-free approximations help processing?

Formalisms describing syntactic structures

Main goal: Describe natural language so a computer can "understand" it.

Important facets

- Linguistic adequacy
- Expressivity
- Scalability
- **Computability**

Context-free grammars I

A CFG is a tuple $\langle T, N, P, S \rangle$ where

- T Terminal symbols
- N Non-terminal symbols
- P Production rules $P : N \rightarrow \alpha$
- S Special start symbol.

T typically model words, N syntactic categories
$\mathscr{L}(CFG)$ is the set of all strings of *terminals* that is reachable by repeated derivation from S

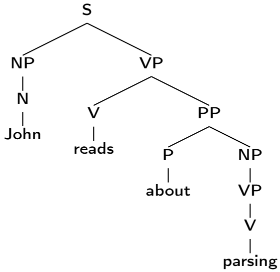
Context-free grammars II

Advantages

- Conceptually easy, well understood
- Very efficient and fast realizations possible
- Parsers scale to huge grammars with thousands of rules and symbols
- Sound probabilistic models exist

Disadvantages

- Linguistically inadequate
- Low expressivity
- Unsuitable for grammar writing



$S \longrightarrow NP\ VP$
$NP \longrightarrow N$
$NP \longrightarrow VP$
$VP \longrightarrow V$
$VP \longrightarrow V\ PP$
$PP \longrightarrow P\ NP$
$N \longrightarrow$ john
$V \longrightarrow$ reads
$V \longrightarrow$ parsing
$P \longrightarrow$ about

Unification-based grammars I

Unification-based grammars extend context-free grammars in two main ways:

- Atomic symbols are replaced with feature structures.
- Production rules are augmented with unification constraints.

Typed unification-based grammars further introduce a type hierarchy.

Unification-based grammars II

Advantages

- Structured objects (FS) and types greatly increase expressivity
- Linguistic adequacy
- Possible to model fine-grained linguistic notions
- Facilitates modeling of (compositional) semantics
- Suitable for grammar engineering, scales to big grammars

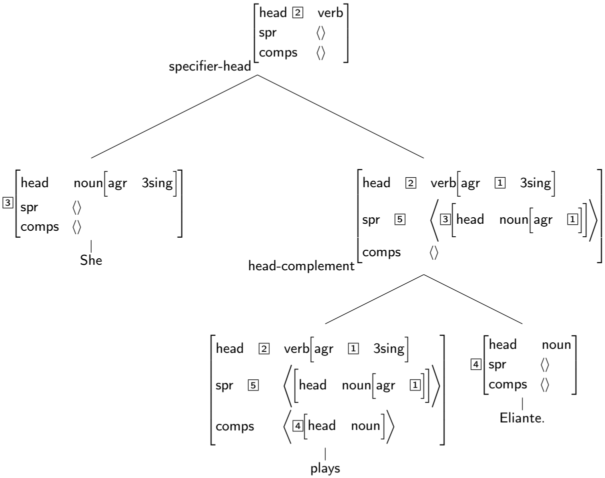
Disadvantages

- Harder to process
- Efficient parsers are conceptually more complex
- Direct stochastical modeling is hard
- Very long processing times, even with highly optimized parsers

HPSG

- Theory of syntax in the generative lexicalist tradition
- Non-transformational, high focus on computability
- Suitable for implementation, several successful implementations
- Grounded in a theory of typed feature structures
- Abstract rule schemata inherits constraints from lexical items

An HPSG derivation



Context-free approximation

General idea

Context-free grammars and unification-based grammars have almost complementary advantages and disadvantages.
Can we synthesize the two approaches and get the best aspects from both?

Approximation

Context-free *approximation* is emerging as a major approach to increasing the practicality of unification-based grammars.

- By "reverting" to the simpler CFG formalism, and trying to describe the original UBG, an *approximation* can be created.
- Automatic process; removes specification burden

Some use-cases of an approximation

Internal

- Recognition filter
- Indirect top-down filtering
- Controlling parser actions

External

- Enable indirect stochastic models
- Aid in supertagging
- **Replace original parser**

Why approximation?

CFGs and UBGs describe languages with different complexity.

	CFG	UBG
Symbols	atomic: $A\ B\ C\quad v, w$	structured: $\mathcal{TFS}$
Productions	$A \rightarrow B\ C \in P$	$A \in P\ \wedge\ A \sqcap B \sqcap C \in \mathcal{TFS}$
Cardinality of symbols	*finite*	*infinite*
Parsing time	*polynomial*	*exponential*

Sound and unsound approximation

Sound approximation

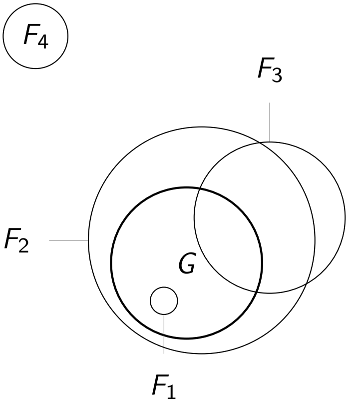
- $\mathscr{L}(G) \subset \mathscr{L}(F_2)$
- “Theoretically clean”
- Safe optimization in some use cases
- Impractical
- Unfeasible on modern grammars?

Unsound approximation

- $\mathscr{L}(G) \not\subseteq \mathscr{L}(F_3)$
- Theoretically unmotivated
- Unsafe optimization
- PCFG modeling
- Tighter approximations possible
- Motivated in more “aggressive” use cases

Properties of approximations

$\mathscr{L}(G) \supset \mathscr{L}(F_1)$
$\mathscr{L}(G) \subset \mathscr{L}(F_2)$
$\mathscr{L}(G) \nsubseteq \mathscr{L}(F_3)$
$\mathscr{L}(G) \cap \mathscr{L}(F_4) = \varnothing$



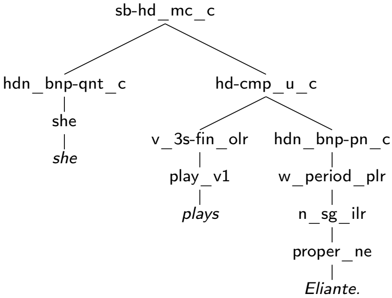
Obtaining an approximation

- Idea: Use already instantiated derivations from original grammar
- If *treebanks* are available, a PCFG can be estimated

Reconstructability

- Special class of approximations enable deterministic replay of an entire derivation, specifying one unique $\mathcal{TFS}$.
- Can recreate all information that was lost to facilitate the approximation.

A Baseline approximation



sb-hd_mc_c	→	hdn_bnp-qnt_c hd-cmp_u_c
hd-cmp_u_c	→	v_3s-fin_olr hdn_bnp-pn_c
v_3s-fin_olr	→	play_v1
n_sg_ilr	→	generic_proper_ne
w_period_plr	→	n_sg_ilr
hdn_bnp-pn_c	→	w_period_plr
hdn_bnp-qnt_c	→	she
she	→	*she*
play_v1	→	*plays*
proper_ne	→	*Eliante.*

Feature Annotation

$$\begin{bmatrix} \textit{hd\_imp\_c} & \\ \text{SYNSEM}\,|\,\text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix}\textit{verb\_full} & \\ \text{AUX} & - \\ \text{VFORM} & \text{imp\_vform}\end{bmatrix} \\ \text{VAL} & \begin{bmatrix}\textit{valence} & \\ \text{SUBJ} & \text{*ocons*} \\ \text{SPR} & \langle\rangle \\ \text{COMPS} & \langle\rangle\end{bmatrix}\end{bmatrix} \\ \text{CONT} & \begin{bmatrix}\textit{mrs} \\ \ldots\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

```
hd_imp_c::[HEAD]verb_full:[SUBJ]*ocons*:[VFORM]imp_vform
```

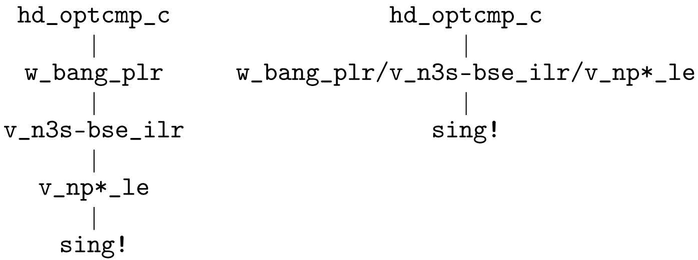
Locality problems in lexical rules

CFGs are local, and the lexical surface is unavailable to any other rule than the preterminals.

- Important in HPSG!
- Feature Annotation doesn't work very well here.
- Naive lexicalization cannot keep track of morphological analysis
- Tree-Rewriting: Collapse each lexical-production chain into one production.

Lexical collapsing

Figure: Original and collapsed parse trees for “Sing!”



Extracting approximations

Implementation

- Implementation in Common Lisp, building on both LKB and [incr tsdb()]
- Flexible system
  - [incr tsdb()] profiles used as input; complex conditions could be used for selection in both training and testing
  - Any LKB/TDL grammar can be approximated
  - Arbitrary tree-collapsing predicates
  - Sanity check for proper CFG
  - Automatically ascertain various measurements
  - Store and read grammars on several formats, including PET PCFG-model and BitPar rule and lexicon files.
- Can *merge* several grammars
- Support parallel extraction

LinGO ERG

- The LinGO ERG is a broad-coverage precision grammar
- Based on HPSG, several man-years of effort
- About 200 syntactic and 100 lexical rule schemata
- Hand-crafted lexicon with 40000 entries, using about 1000 *lexical types*

A note on token normalization

- ERG uses pseudo-morphological affixes for punctuation
- Morphological component difficult to map directly to CFG
- Lattice-based input
- Simulation using lexical production yield
- Remove rules pertaining to punctuation

Extraction experiments I

- WeScience treebank
  - Wikipedia articles
  - Grammar-supported treebank
  - Derivations manually disambiguated parse-results
  - Sections 1-12 used for training
  - Section 13 used for testing
- Static measures from the definitions of the grammar
- Dynamic measures by measuring on a held-out corpus
- Can serve as guidelines for finding a starting point when applying approximations to new use-cases

Extraction experiments II

Grammars with varying levels of annotation (A) and how they overlap WS13

A	W	N	P	W_h	N_h	P_h	C	PC	TC
0	20245	886	8650	3782	557	2791	100%	90%	**89%**
1	20245	1280	9921	3782	725	3021	100%	89%	73%
10	20245	3600	20815	3782	1593	5085	97%	81%	52%

Grammars with *lexical collapsing* and token normalization and how they overlap WS13

A	W	N	P	W_h	N_h	P_h	C	PC	TC
0-LC	21962	1583	9213	3989	767	2640	98%	87%	**68%**
1-LC	21962	1752	10032	3989	848	2806	98%	86%	66%
10-LC	21962	3437	19698	3989	1533	4620	86%	79%	48%

Extraction experiments III

WikiWoods

- Automatically disambiguated treebank of the *entire* English Wikipedia as of 2008
- About 55 million sentences, 48 with a usable parse
- Noisy data
- Small sample of ~450000 sentences used here

d	A	W	N	P	W_h	N_h	P_h	C	PC	TC
WS	10	21962	3437	19698	3989	1533	4620	86%	79%	48%
WS	33	21962	8283	34472	3989	2825	6383	46%	69%	34%
WW	10	459244	9105	163652	3989	1533	4620	97%	94%	80%
WW	33	459244	39962	406760	3989	2810	6333	97%	92%	59%

Standalone parsing

Parsing task

- Classical PCFG parser explores the search space
- Most “aggressive” use of approximations?
- BitPar; off-the-shelf high-performance parser
- ParsEval and exact-match metrics

Parsing experiments I

A	C%	P%	R%	F1	EX%	TA%	μ	Σ
PET	100	84	84	84	46	96	3.6k	2846
0	100	65	59	62	11	89	385	303
1	100	71	64	68	12	89	115	90
10	97	77	73	75	20	91	42	33
0-LC	98	73	72	72	23	86	128	100
1-LC	98	77	77	77	23	87	36	28
10-LC	86	81	81	81	32	92	24	19

Lexical collapsing

- Large boost in accuracy *and* processing time in all configurations
- A “necessary” strategy in successful stand-alone parsing
- Coverage problems

Parsing experiments II

A	C%	P%	R%	F1	EX%	TA%	μ	Σ
PET	100	87	87	87	46	96	3.6k	2846
WS-10	86	81	81	81	32	92	24	19
WS-33	46	87	88	88	61	96	47	37
WW-10	97	83	84	84	32	92	329	259
WW-33	97	83	83	83	37	92	571	449

- Can trade accuracy for coverage
- Using noisy data can increase performance
- Processing times no longer as encouraging

Parsing algorithms

- BitPar uses bit-fields internally
- Can parallelize bottom-up parsing elegantly
- Not suitable for all types of grammars

Our parser

- Implemented in Common Lisp
- Augmented CKY-style parser that handles unary rules
- Full forest construction
- Viterbi and *n*-best output possible

A	C%	P%	R%	F1	EX%	TA%	μ	Σ
PET	100	87	87	87	46	96	3.6k	2846
BitPar-33	97	83	83	83	37	92	571	449
CKY+-33	97	83	83	83	37	92	142	112

Meta-comparison of parsers

- Stand-alone parsing is a trending topic
- CuteForce (Ytrestøl 2011), SR-style Oracle guided (uses supertagged input)
- Zhang and Krieger (2011), BitPar-style parser, grandparenting techniques

A	C%	P%	R%	F1	EX%	TA%	μ	Σ
PET	100	87	87	87	46	96	3.6k	2846
CKY+-33	97	83	83	83	37	92	142	112
CuteForce	99	-	-	82	36	95	15	-
Zhang and Krieger (2011)	-	80	79	80	32	93	-	-

Concluding remarks

Concluding remarks I

Approximation

- Context-free approximation can serve to increase the practicality of precision grammars
- The reconstructable approximations are of particular interest
- While the syntactic parts of precision grammars can be approximated intuitively, morphological engines can create problems

Concluding remarks II

Stand-alone parsing

- Initial experiments promising, high-coverage, fast grammar can readily be obtained for a slight drop in accuracy
- High levels of internal annotation can give equal performance to external annotation
- Bit-field optimization not necessarily always given

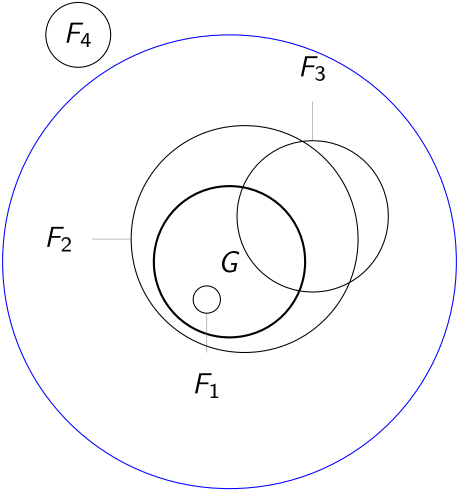
Outlook I

- High-quality supertagging probably beneficial
- Integrating original parse-selection component
- Better evaluation systems, move away from syntacto-centric evaluation
- Use type-hierarchy actively to combat sparseness
- Take advantage of lattice based input and only approximate the syntactic part of the grammar

Outlook II

- "Middle" Natural language processing?
- PCFG Language modeling
- Relationship of approximation to split-merge techniques

Properties of approximations



Thank you