Constraining robust constructions for broad-coverage parsing with precision grammars

Bart Cramer (joint work with Yi Zhang)

Department of Computational Linguistics & Phonetics, Saarland University
bcramer@coli.uni-saarland.de

July 2010, DELPH-IN summit, Paris

Cheetah

Cramer & Zhang, 2009

The Cheetah grammar for German consists of two components:

- A hand-written core grammar
  - The structure is mainly inspired by Hinrichs & Nakazawa (1994), Müller (2002), and Crysmann (2003; 2005). Some interesting phenomena are covered: Mittelfeld scrambling; extraposition of complements, adjuncts and relative clauses; certain forms of ellipsis.
  - The grammar contains 89 phrasal rules (of which 42 are for coordinations), and 14 lexical rules.
  - A core lexicon is included for closed word classes: auxiliary verbs, pronouns, determiners, etc. It contains 546 lemmas.

Cheetah

Cramer & Zhang, 2009

- An automatically derived lexicon
  - 90% of the Tiger treebank (Brants et al., 2002) is used to learn lexical entries from.
  - The (deterministic; heuristic) algorithm maps lemmas to fairly detailed lexical types (e.g 200 verbal lexical types are found).
  - Morphology is handled as if verbs/nouns/adjectives are irregular, listing (lemma, inflection, word form) triples.

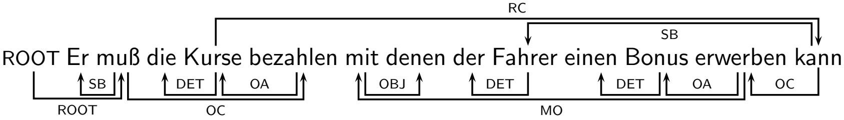
Cheetah

Cramer & Zhang, 2009

- An automatically derived lexicon
  - 90% of the Tiger treebank (Brants et al., 2002) is used to learn lexical entries from.
  - The (deterministic; heuristic) algorithm maps lemmas to fairly detailed lexical types (e.g 200 verbal lexical types are found).
  - Morphology is handled as if verbs/nouns/adjectives are irregular, listing (lemma, inflection, word form) triples.

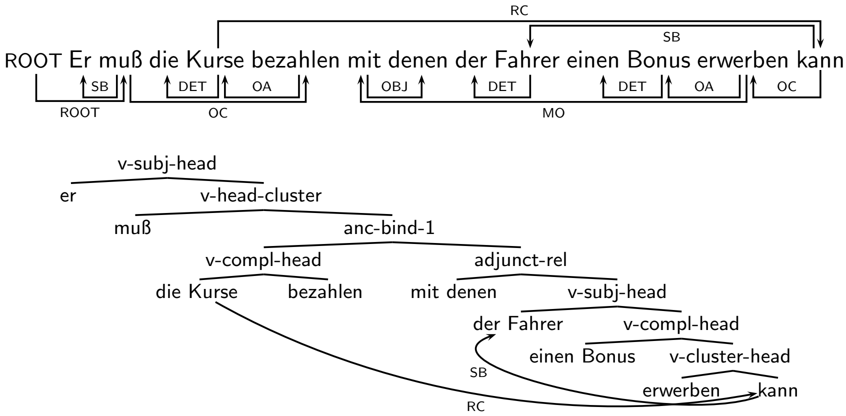
Part-of-speech	Lemmas	Lexical entries	Inflection triples
Verbs	4543	9235	18745
Nouns	33835	34821	51303
Names	12445	12783	na
Adjectives	7318	8018	50480
Adverbs	2654	4577	na

Cheetah



Lit.: He has-to the course pay with which the driver a bonus acquire can.

Cheetah



Lit.: He has-to the course pay with which the driver a bonus acquire can.

Semantics

```
[ LTOP: h1
  INDEX: e2
  RELS: <
          [ "_det_der_DET_rel"
            LBL: h3
            ARG0: x5
            ARG1: x4 ]
          [ "_noun_story__rel"
            LBL: h6
            ARG0: x4 ]
          [ "_v_gehen_SB_rel"
            LBL: h7
            ARG0: e2
            ARG1: x4 ]
          [ "_adv_so_MO_rel"
            LBL: h8
            ARG0: e9
            ARG1: e2 ] >
  HCONS: < > ]
```

Semantics

```
[ LTOP: h1
  INDEX: e2
  RELS: <
          [ "_det_der_DET_rel"
            LBL: h3
            ARG0: x5
            ARG1: x4 ]
          [ "_noun_story__rel"
            LBL: h6
            ARG0: x4 ]
          [ "_v_gehen_SB_rel"
            LBL: h7
            ARG0: e2
            ARG1: x4 ]
          [ "_adv_so_MO_rel"
            LBL: h8
            ARG0: e9
            ARG1: e2 ] >
  HCONS: < > ]
```

ROOT	ROOT	geht
so	MO	geht
geht	SB	story
der	DET	story

After reversal of the DET and modifier labels and removal of non-lexical relations, this yields the following:

ROOT	ROOT	geht
geht	MO	so
geht	SB	story
story	DET	der

The Savannah treebank

- The Savannah treebank is created as follows:
  - Parse the raw text and record the parse trees, including the MRSs.
  - For each sentence, convert all readings' MRSs to dependencies.
  - If the best reading's f-score is higher than a threshold β, that reading is accepted; otherwise, all readings are rejected.
- Around 55% of the sentences receive a good analysis (f-score higher than 0.9), resulting in a tsdb treebank with 25k trees.

The Savannah treebank

- The Savannah treebank is created as follows:
  - Parse the raw text and record the parse trees, including the MRSs.
  - For each sentence, convert all readings' MRSs to dependencies.
  - If the best reading's f-score is higher than a threshold β, that reading is accepted; otherwise, all readings are rejected.
- Around 55% of the sentences receive a good analysis (f-score higher than 0.9), resulting in a tsdb treebank with 25k trees.
- This mechanism also allows for unit testing: do DLA on one sentence; parse that sentence; can the original dependencies be discovered?

Introduction

Why restricting the search space?

- Not only ‘just’ to be faster.

Introduction

Why restricting the search space?

- Not only ‘just’ to be faster.
- Sentences that timeout can be pulled inside the timeout window, extending the coverage.

Phrasal restriction

- Ninomiya et al. (2005) describe how pruning can make the Enju HPSG parser for English more efficient.
  - They use a local discriminative model to rank all chart items within one chart cell, and remove those that had much lower figures of merit than the best item in the cell.
  - The best results were obtained by iterative parsing, slowly widening the bandwidth until a parse is found.

Phrasal restriction

- Ninomiya et al. (2005) describe how pruning can make the Enju HPSG parser for English more efficient.
    - They use a local discriminative model to rank all chart items within one chart cell, and remove those that had much lower figures of merit than the best item in the cell.
    - The best results were obtained by iterative parsing, slowly widening the bandwidth until a parse is found.
- Cahill et al. (2008) report on a similar approach, pruning the c-structures of the XLE LFG parser for English.
    - The main differences: the figures of merit are based on a generative model; expensive unifications can be prevented, because the f-structures are only computed after the parse forest is ready. A speedup of 67% was reported, with a slight increase in f-score.

The PET parser

- We propose a model based on the generative probabilities on the HPSG rule applications
- Instead of choosing a certain bandwidth, our method keeps the size of the cell fixed.
- The algorithm will alter the agenda, an important element in the PET parser, which is implemented as a priority queue.

Agenda

rule+passive

unary

$$\begin{array}{ccccc} R & & & & R \\ | & + & P & \Rightarrow & | \\ & & & & P \end{array}$$

binary

$$\begin{array}{ccccccccc} & R & & & & & & R & \\ / & & \backslash & + & P & \Rightarrow & / & & \backslash \\ & & & & & & P & & \end{array}$$

active+passive

binary

$$\begin{array}{ccccccccc} & R & & & & & & R & \\ / & & \backslash & + & P_2 & \Rightarrow & / & & \backslash \\ P_1 & & & & & & P_1 & & P_2 \end{array}$$

Agenda

Each time a task succeeds, the following happens:

- For each inserted passive item, add (rule+passive) tasks that combine the passive item with each of the rules, and add (active+passive) tasks that combine with each of the neighbouring active items.
- For each inserted active item, add (active+passive) tasks that combine the remaining gaps in the active item with existing neighbouring passive items in the chart.

Agenda

Each time a task succeeds, the following happens:

- For each inserted passive item, add (rule+passive) tasks that combine the passive item with each of the rules, and add (active+passive) tasks that combine with each of the neighbouring active items.
- For each inserted active item, add (active+passive) tasks that combine the remaining gaps in the active item with existing neighbouring passive items in the chart.

So: each created chart item spawns new tasks, and successful tasks/unifications create new chart items. This process continues until no tasks are left on the agenda, after which the solutions are harvested from the chart.

Defining the priorities

The generative model computes the probability P_r of the resulting passive item.

Defining the priorities

The generative model computes the probability P_r of the resulting passive item.

Task	Previous		Conditional
Active + passive, binary	$p(P_1) \cdot p(P_2)$	$\cdot$	$p(R \rightarrow P_1 P_2)$
Rule + passive, unary	$p(P)$	$\cdot$	$p(R \rightarrow P)$
Rule + passive, binary	$p(P_1) \cdot p(?)$	$\cdot$	$p(R \rightarrow P_1 ?)$

- In the last case, the probability of the resulting passive chart item can't be computed. As a workaround, we set the missing probabilities to 1.

Defining the priorities

The generative model computes the probability P_r of the resulting passive item.

Task	Previous		Conditional
Active + passive, binary	$p(P_1) \cdot p(P_2)$	$\cdot$	$p(R \rightarrow P_1 P_2)$
Rule + passive, unary	$p(P)$	$\cdot$	$p(R \rightarrow P)$
Rule + passive, binary	$p(P_1) \cdot p(?)$	$\cdot$	$p(R \rightarrow P_1 ?)$

- In the last case, the probability of the resulting passive chart item can't be computed. As a workaround, we set the missing probabilities to 1.
- To differentiate between likely and less likely rules, the priorities are defined as follows: $Pr = p(R)p(P_r)$

Search space restriction

- The number of tasks is restricted on a local level: a maximum number of tasks is defined for each span (i, j).
- We define three different strategies:

  All: All tasks are counted

  Success: Only successful tasks are counted (that is: if the unification succeeds)

  Passive: Only those successful tasks are counted that lead to a passive item
- Morphological and lexical rule applications are not counted, and hence not restricted. Phrasal unary rules are counted.

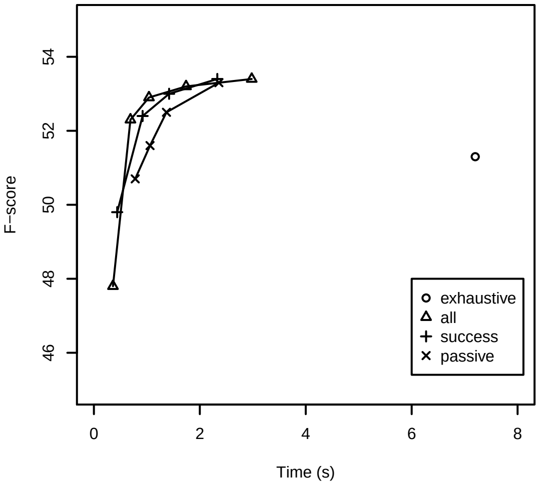
Experimental set-up

- A generative model (for the priorities) and a discriminative model (for parse disambiguation) were trained from the HPSG treebank (25k trees).
- We extracted the text and the gold standard syntactic dependencies from the Tiger treebank, sentences s47500-s50000.
- The text was parsed using Cheetah, and the dependencies from the output were compared to the gold standard.
- Maximum parsing time was set to 60 seconds, after which solutions were extracted from the parse forest created so far.

Results

Strategy	exhaustive	all	success	passive
Cell size		3000	200	100
Time (s)	7.20	1.04	0.92	1.06
Coverage	59.4%	60.5%	60.0%	59.0%
Exact	17.6%	17.6%	17.4%	17.4%
Recall	37.6%	39.5%	38.9%	38.0%
Precision	80.7%	80.3%	80.1%	80.4%
F-score	51.3%	52.9%	52.4%	51.6%

Results


54
52
50
48
46
F−score
0
2
4
6
8
Time (s)
exhaustive
all
success
passive

Results

- The average parsing time can be reduced by $> 80\%$...,
- ... retaining the parser's precision ...,
- ... and a slight increase in coverage ($< 1\%$).
- The time/quality trade-offs are very similar for the three strategies (all, success, passive).

In practice

This functionality has been integrated into the chart mapping branch of PET lately. The following steps are necessary to reproduce this behaviour:

- Learn a `.gm` model, using the Python script I put online together with the slides. The only thing that is needed is a tsdb treebank. Computation time is in the order of minutes.
- Add the following line to your `.set` file:
  `gm := "yourmodel.gm".`
- Make use of the `-local-cap=size` and `-count-tasks=(0,1,2)` options in the `cheap` command.

Enhancing robustness

- Heavily constrained unification grammars allow for a linguistically interesting grammar, but also causes low coverage.
- Possible solutions:
  - Remove constraints from the grammar, allowing for more overgeneration.
  - Mine the chart to extract a fragment analysis (Riezler et al., 2001; Kiefer et al., 1999, Zhang et al., 2007)

Enhancing robustness

- Heavily constrained unification grammars allow for a linguistically interesting grammar, but also causes low coverage.
- Possible solutions:
  - Remove constraints from the grammar, allowing for more overgeneration.
  - Mine the chart to extract a fragment analysis (Riezler et al., 2001; Kiefer et al., 1999, Zhang et al., 2007)
- Instead, we use overgenerating robustness rules (RRs) to parse extra-grammatical sentences.

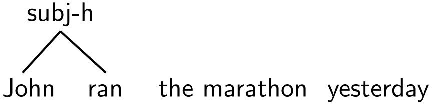
Robustness rules

Let's assume that the grammar only lists 'to run' as an intransitive verb.

John ran the marathon yesterday

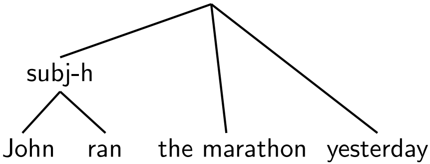
Robustness rules

Let's assume that the grammar only lists ‘to run’ as an intransitive verb.



Robustness rules

Let's assume that the grammar only lists 'to run' as an intransitive verb.

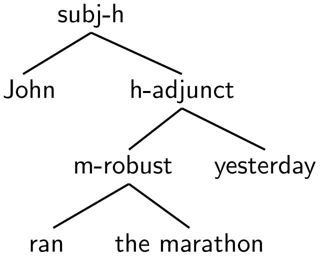


Robustness rules

It would be more desirable to overcome this barrier on a lower level, localising the damage:

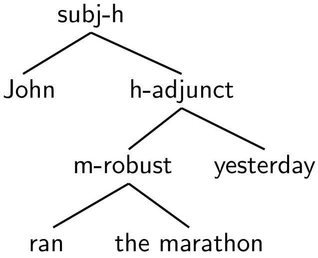
Robustness rules

It would be more desirable to overcome this barrier on a lower level, localising the damage:



Robustness rules

It would be more desirable to overcome this barrier on a lower level, localising the damage:



The advantage: the dependency between ‘ran’ and ‘yesterday’ is recovered.

Robustness rules

- If the RRs produce general features structure, the packing mechanism is influenced heavily:
  - All non-robust chart items are more specific than their robust siblings, and will hence be packed.
  - This leads to a very compact chart, but unpacking solutions will lead to many unfications failures (and impermissible unpacking times).
- Therefore, the level of constriction (on the feature level) must be retained somehow.

Robustness rules

$$\begin{bmatrix} \text{structure-robust} & \\ \text{SYNSEM} & \boxed{1} \\ \text{ROBUST} & + \\ \text{MN-DTR} & \begin{bmatrix} \text{sign} & \\ \text{SYNSEM} & \boxed{1}\begin{bmatrix} \text{LOCAL.CAT.HEAD} & \text{verb} \end{bmatrix} \\ \text{ROBUST} & \text{-} \end{bmatrix} \\ \text{RB-DTR} & \begin{bmatrix} \text{sign} & \\ \text{SYNSEM} & \begin{bmatrix} \text{NONLOCAL} & \text{no-nonlocal} \end{bmatrix} \\ \text{ROBUST} & \text{-} \end{bmatrix} \end{bmatrix}$$

Robustness rules

- Two robustness rules pairs were added to the grammar:
  - **+V** The robust daughter is a verb, which is still allowed to have valence, but cannot have any features in NONLOCAL.
  - **+NV** The robust daughter is anything but a verb, cannot have any non-empty valence list, and cannot have any features in NONLOCAL.
- Robustness rules do not contribute a dependency.

Robustness rules

- During parse forest creation:
  - Application of RRs is discouraged by adding a large penalty to the task's priority.
  - That means that first a chart is built using the standard set of rules.
  - Chart cells that haven't been filled with items from the standard grammar will receive additional attention using the RRs.

Robustness rules

- During parse forest creation:
  - Application of RRs is discouraged by adding a large penalty to the task's priority.
  - That means that first a chart is built using the standard set of rules.
  - Chart cells that haven't been filled with items from the standard grammar will receive additional attention using the RRs.
- During unpacking:
  - The application of RRs is strongly dispreferred by the disambiguation model.
  - Hence, sentences that would be fine with the standard grammar remain uncompromised.
  - All solutions with an equal number of RR applications retain their relative order, so the disambiguation model can still identify the best solution.

Results

Different robustness rules, success-200 strategy

		standard		+V	+NV	+V+NV
		exhaustive	restricted		restricted	
	time (s)	7.20	0.92	4.10	1.42	4.09
no fragment	coverage	59.3%	60.0%	72.6%	69.9%	78.6%
	recall	37.6%	38.9%	48.4%	47.0%	53.8%
	precision	80.7%	80.1%	78.6%	78.2%	77.7%
	f-score	51.3%	52.4%	59.9%	58.7%	63.6%

Results

- The use of robustness rules +V and +NV increase coverage by 13% and 10% respectively. The combination of both yields a 19% increase.
- For +NV, the time penalty is small (0.5s), whereas it is acceptable for both +V and +V+NV (3.2s). However, +V+NV with parse restriction is still 43% faster than the standard grammar.
- The robustness rules have a modest negative impact on the precision of the parser (3% for +V+NV).

Results

Different robustness rules, success-200 strategy

		standard		+V	+NV	+V+NV
		exhaustive	restricted	restricted		
	time (s)	7.20	0.92	4.10	1.42	4.09
no fragment	coverage	59.3%	60.0%	72.6%	69.9%	78.6%
	recall	37.6%	38.9%	48.4%	47.0%	53.8%
	precision	80.7%	80.1%	78.6%	78.2%	77.7%
	f-score	51.3%	52.4%	59.9%	58.7%	63.6%
fragment	coverage	94.3%	98.3%	98.5%	98.7%	98.5%
	recall	50.4%	53.6%	59.5%	56.9%	61.3%
	precision	75.4%	75.0%	75.0%	74.5%	74.7%
	f-score	60.4%	62.5%	66.3%	64.5%	67.3%

Results

+V+NV, different strategies

		all-3000	success-200	passive-100
	time (s)	4.18	4.09	5.58
no fragment	coverage	72.0%	78.6%	72.6%
	recall	47.3%	53.8%	48.4%
	precision	78.5%	77.7%	78.6%
	f-score	59.0%	63.6%	59.9%
fragment	coverage	98.0%	98.5%	97.6%
	recall	60.1%	61.3%	59.9%
	precision	74.4%	74.7%	74.2%
	f-score	66.5%	67.3%	66.3%

Results

- Using fragment analyses as a fallback strategy makes the parser's coverage approximate 100%.
- The combination of robustness rules and fragment analyses perform significantly better (5%) than just using fragment analyses.
- Put under more pressure than in the restriction experiments, the success strategy offers a better time/coverage tradeoff than the all and passive strategies.

In practice

The following steps are needed to make the robustness rules work in practice:

- Follow the instructions on search restriction.
- Add the RRs to the rules file.
- Add the following to your `.set` file, in order to give these rule applications lower priority and disambiguation scores:
  `robust-rules := $rr1 $rr2.`

Conclusion

- In the restriction experiments, the same trends as in Cahill et al. (2008) were observed: large speedups, no loss of precision, with a small increase of coverage/f-score.
- Carefully engineered robustness rules in combination with a per-cell cap on the number of successful tasks forms an atractive strategy to increase coverage of precision grammars.

Conclusion

- In the restriction experiments, the same trends as in Cahill et al. (2008) were observed: large speedups, no loss of precision, with a small increase of coverage/f-score.
- Carefully engineered robustness rules in combination with a per-cell cap on the number of successful tasks forms an atractive strategy to increase coverage of precision grammars.
- Future work consists of finding statistically more sound ways to estimate the probabilities for robustness rules.
    - A possible advantage is that the generative model will be better able to identify where and how to patch.
    - The model might learn that one RR application is better than a really awkward solution from the standard grammar.