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

12 May 2010, FEAST, Saarbrücken

Introduction

- When parsing with deep grammars (grammars embedded in a linguistic theory, such as LFG or HPSG), there is a trade-off between efficiency, coverage and accuracy.

Introduction

- When parsing with deep grammars (grammars embedded in a linguistic theory, such as LFG or HPSG), there is a trade-off between efficiency, coverage and accuracy.
- In particular, parsers based on precision grammars (deep grammars that also aim to reject ungrammatical sentences) face the problem of low coverage.

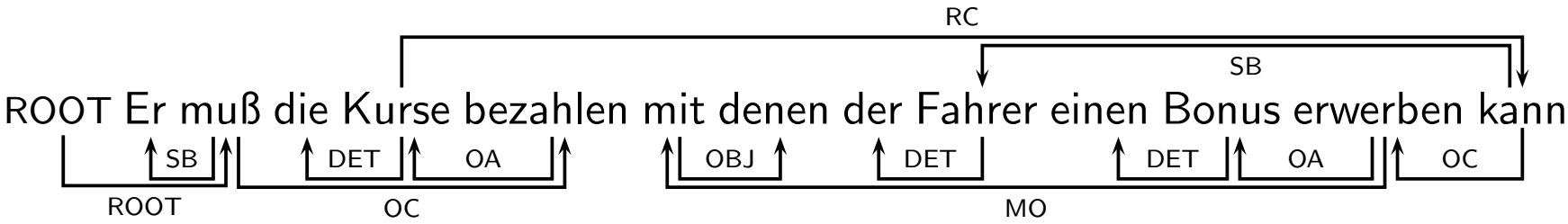
Introduction

- When parsing with deep grammars (grammars embedded in a linguistic theory, such as LFG or HPSG), there is a trade-off between efficiency, coverage and accuracy.
- In particular, parsers based on precision grammars (deep grammars that also aim to reject ungrammatical sentences) face the problem of low coverage.
- In this study, we will propose a method that addresses efficiency and robustness concerns concurrently.

Cheetah

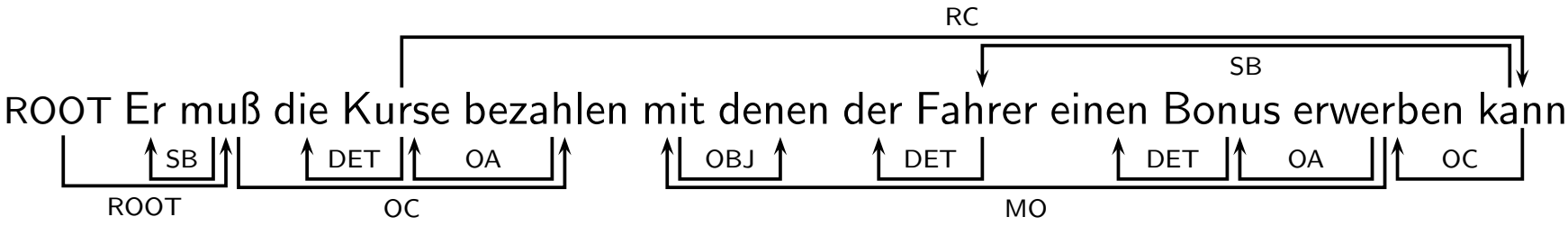
- All experiments in this study are carried out with Cheetah, a medium-sized HPSG precision grammar for German (Cramer & Zhang, 2009).
- It is the combination of a hand-written core grammar (including a core lexicon for syntactically idiosyncratic words) and an open word class lexicon extracted from 90% of the Tiger treebank.
- The core grammar covers a few interesting phenomena: Mittelfeld scrambling; extraposition of complements, adjuncts and relative clauses; certain forms of ellipsis.
- Around 25k sentences of the Tiger treebank is converted to HPSG derivation trees. From this treebank, a MaxEnt disambiguation model has been trained.

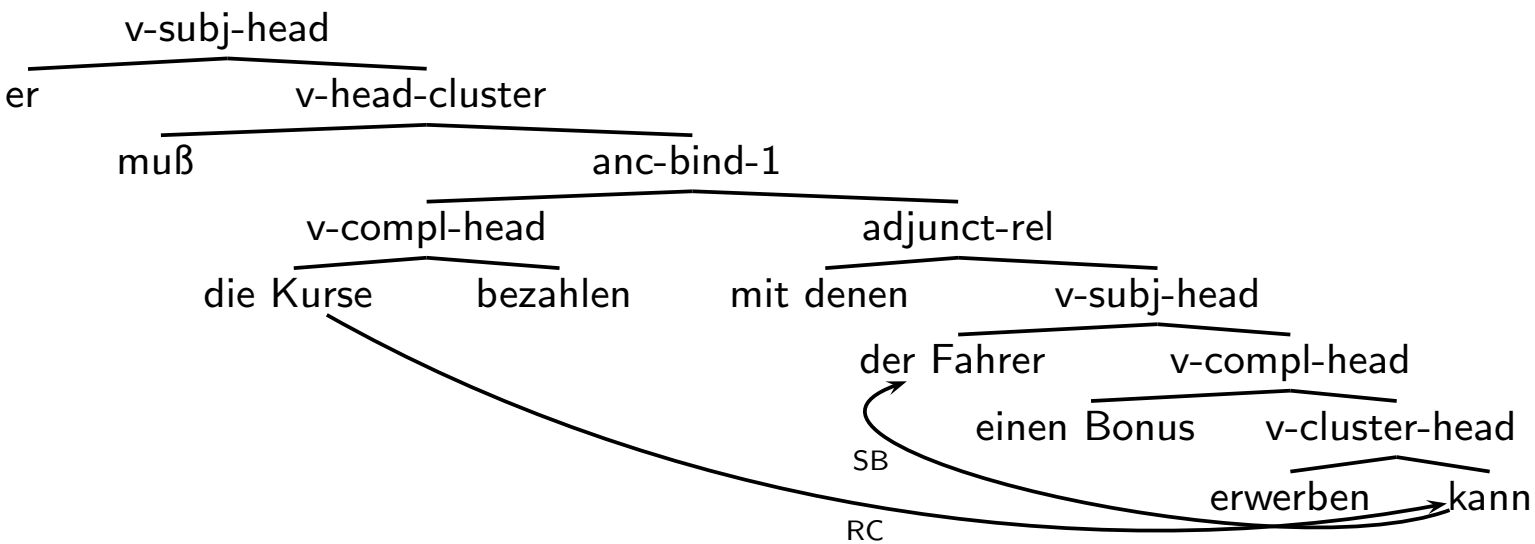
Cheetah



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.

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.

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.
- 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.

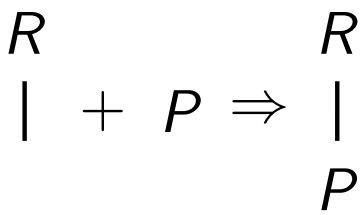
The PET parser

- The PET parser (Callmeier, 2000) is a unification-based, bottom-up parser for HPSG, implemented in C/C++.
- Typical grammars used with this parser use large features structures to represent linguistic information.
- It contains several optimisation techniques, such as quasi-destructive unification, hyper-active parsing, subsumption-based packing (Oepen & Carroll, 2000) and selective unpacking (Zhang et al., 2007).
- One of its central data structures is the agenda, implemented as a priority queue.

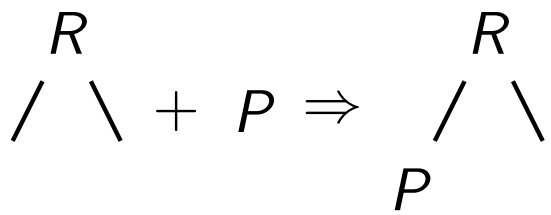
Agenda

rule+passive

unary

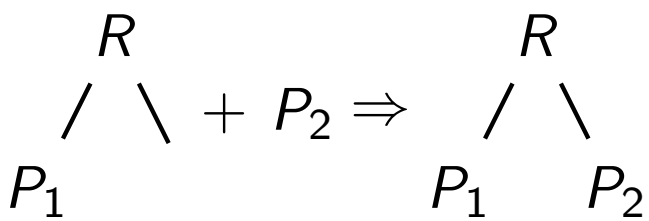


binary



active+passive

binary



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.

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
Successful	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.

Defining the priorities

- Probabilities of the chart items are based on a generative model of rule applications. Lidstone smoothing is applied to estimate the probability of unseen subtrees.

Defining the priorities

- Probabilities of the chart items are based on a generative model of rule applications. Lidstone smoothing is applied to estimate the probability of unseen subtrees.
- The priorities of the tasks are calculated according to the following formulae:
  - rule+passive (unary & binary):
    $Pr = p(R) \cdot p(P)$
  - active+passive (binary):
    $Pr = p(R) \cdot p(P_1) \cdot p(P_2)$

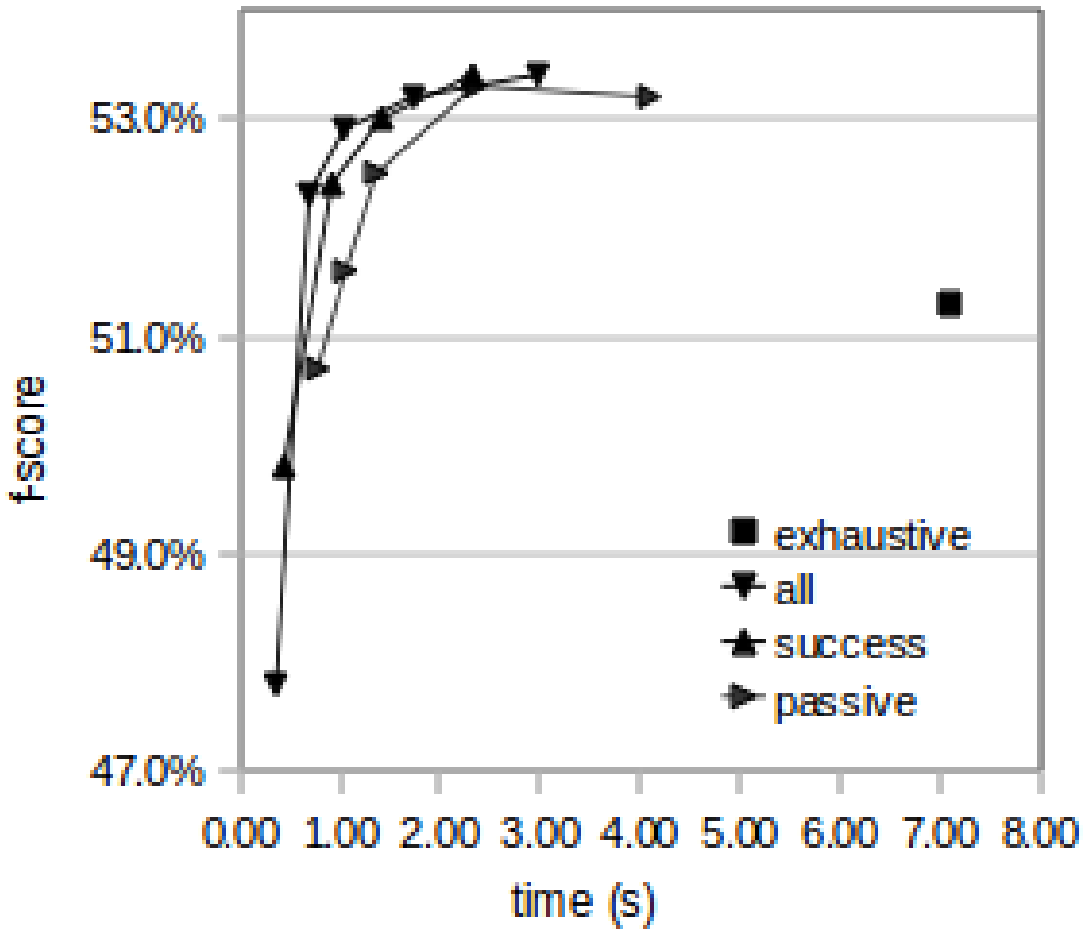
Experimental set-up

- 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


53.0%
51.0%
49.0%
47.0%
fscore
0.00 1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00
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, successful, passive).

Robustness rules

- The heavily constrained unification grammar allows 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 (e.g. Riezler et al., 2001; Kiefer et al., 1999)

Robustness rules

- The heavily constrained unification grammar allows 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 (e.g. Riezler et al., 2001; Kiefer et al., 1999)
- Instead, we use overgenerating robustness rules to parse extra-grammatical sentences.

Robustness rules

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

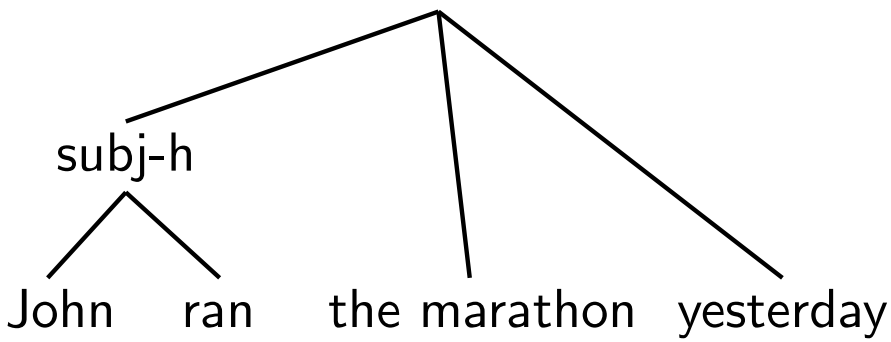
John ran the marathon yesterday

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.

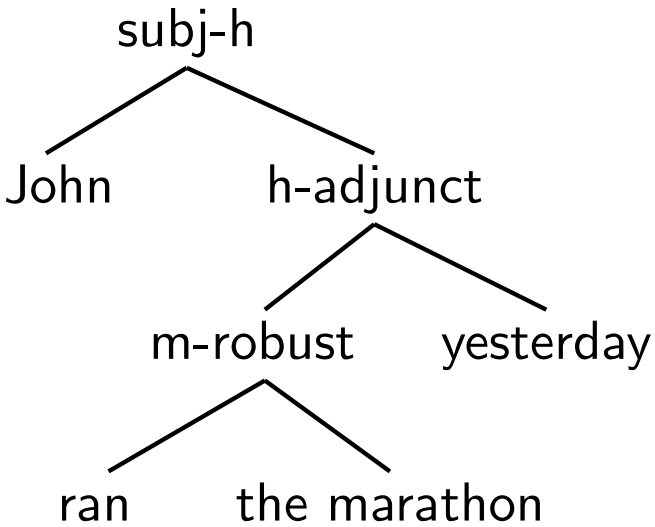


Robustness rules

It would be more desirable to overcome this barrier on a lower level:

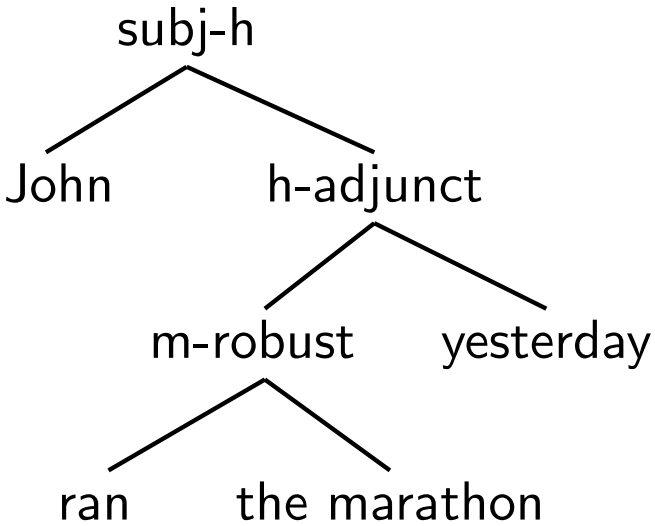
Robustness rules

It would be more desirable to overcome this barrier on a lower level:



Robustness rules

It would be more desirable to overcome this barrier on a lower level:



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

Robustness rules

Robustness rules

- If after the application of a robustness rule the chart item is highly underspecified, this will lead to an intractable search space (and interference with the packing mechanism).

Robustness rules

- If after the application of a robustness rule the chart item is highly underspecified, this will lead to an intractable search space (and interference with the packing mechanism).
- Therefore, the robustness rules must retain the linguistic depth featured in its MAIN daughter:

$$\left[\begin{array}{ll} \textit{structure-robust} & \\ \text{SYNSEM} & \boxed{1} \\ \text{ROBUST} & + \\ \text{MN-DTR} & \left[\begin{array}{ll} \textit{sign} & \\ \text{SYNSEM} & \boxed{1}\left[\text{LOCAL.CAT.HEAD verb}\right] \\ \text{ROBUST} & - \end{array}\right] \\ \text{RB-DTR} & \left[\begin{array}{ll} \textit{sign} & \\ \text{SYNSEM} & \left[\text{NONLOCAL no-nonlocal}\right] \\ \text{ROBUST} & - \end{array}\right] \end{array}\right]$$

Robustness rules

- Two robustness rules 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, successful-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, successful-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	successful-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 successful strategy offers a better time/coverage tradeoff than the all and passive strategies.

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.