Integrated Shallow and Deep Parsing: TopP meets HPSG

Anette Frank, Markus Becker[‡], Berthold Crysmann, Bernd Kiefer and Ulrich Schäfer

DFKI GmbH
66123 Saarbrücken, Germany
firstname.lastname@dfki.de

School of Informatics[‡]
University of Edinburgh, UK
M.Becker@ed.ac.uk

Abstract

We present a novel, data-driven method for integrated shallow and deep parsing. Mediated by an XML-based multi-layer annotation architecture, we interleave a robust, but accurate stochastic topological field parser of German with a constraint-based HPSG parser. Our annotation-based method for dovetailing shallow and deep phrasal constraints is highly flexible, allowing targeted and fine-grained guidance of constraint-based parsing. We conduct systematic experiments that demonstrate substantial performance gains.[1]

1 Introduction

One of the strong points of deep processing (DNLP) technology such as HPSG or LFG parsers certainly lies with the high degree of precision as well as detailed linguistic analysis these systems are able to deliver. Although considerable progress has been made in the area of processing speed, DNLP systems still cannot rival shallow and medium depth technologies in terms of throughput and robustness. As a net effect, the impact of deep parsing technology on application-oriented NLP is still fairly limited.

With the advent of XML-based hybrid shallow-deep architectures as presented in (Grover and Lascarides, 2001; Crysmann et al., 2002; Uszkoreit, 2002) it has become possible to integrate the added value of deep processing with the performance and robustness of shallow processing. So far, integration has largely focused on the lexical level, to improve upon the most urgent needs in increasing the robustness and coverage of deep parsing systems, namely lexical coverage. While integration in (Grover and Lascarides, 2001) was still restricted to morphological and PoS information, (Crysmann et al., 2002) extended shallow-deep integration at the lexical level to lexico-semantic information, and named entity expressions, including multiword expressions.

(Crysmann et al., 2002) assume a vertical, 'pipeline' scenario where shallow NLP tools provide XML annotations that are used by the DNLP system as a preprocessing and lexical interface. The perspective opened up by a multi-layered, data-centric architecture is, however, much broader, in that it encourages horizontal cross-fertilisation effects among complementary and/or competing components.

One of the culprits for the relative inefficiency of DNLP parsers is the high degree of ambiguity found in large-scale grammars, which can often only be resolved within a larger syntactic domain. Within a hybrid shallow-deep platform one can take advantage of partial knowledge provided by shallow parsers to pre-structure the search space of the deep parser. In this paper, we will thus complement the efforts made on the lexical side by integration at the phrasal level. We will show that this may lead to considerable performance increase for the DNLP component. More specifically, we combine a probabilistic topological field parser for German (Becker and Frank, 2002) with the HPSG parser of (Callmeier, 2000). The HPSG grammar used is the one originally developed by (Müller and Kasper, 2000), with significant performance enhancements by B. Crysmann.

In Section 2 we discuss the mapping problem involved with syntactic integration of shallow and deep analyses and motivate our choice to combine the HPSG system with a topological parser. Section 3 outlines our basic approach towards syntactic shallow-deep integration. Section 4 introduces various confidence measures, to be used for fine-tuning of phrasal integration. Sections 5 and 6 report on

[1]This work was in part supported by a BMBF grant to the DFKI project WHITEBOARD (FKZ 01 IW 002).

experiments and results of integrated shallow-deep parsing, measuring the effect of various integration parameters on performance gains for the DNLP component. Section 7 concludes and discusses possible extensions, to address robustness issues.

2 Integrated Shallow and Deep Processing

The prime motivation for integrated shallow-deep processing is to combine the robustness and efficiency of shallow processing with the accuracy and fine-grainedness of deep processing. Shallow analyses could be used to pre-structure the search space of a deep parser, enhancing its efficiency. Even if deep analysis fails, shallow analysis could act as a guide to select partial analyses from the deep parser's chart – enhancing the robustness of deep analysis, and the informativeness of the combined system.

In this paper, we concentrate on the usage of shallow information to increase the efficiency, and potentially the quality, of HPSG parsing. In particular, we want to use analyses delivered by an efficient shallow parser to pre-structure the search space of HPSG parsing, thereby enhancing its efficiency, and guiding deep parsing towards a best-first analysis suggested by shallow analysis constraints.

The search space of an HPSG chart parser can be effectively constrained by external knowledge sources if these deliver compatible partial subtrees, which would then only need to be checked for compatibility with constituents derived in deep parsing. Raw constituent span information can be used to guide the parsing process by penalizing constituents which are incompatible with the precomputed 'shape'. Additional information about proposed constituents, such as categorial or featural constraints, provide further criteria for prioritising compatible, and penalising incompatible constituents in the deep parser's chart.

An obvious challenge for our approach is thus to identify suitable shallow knowledge sources that can deliver compatible constraints for HPSG parsing.

2.1 The Shallow-Deep Mapping Problem

However, chunks delivered by state-of-the-art shallow parsers are not isomorphic to deep syntactic analyses that explicitly encode phrasal embedding structures. As a consequence, the boundaries of deep grammar constituents in (1.a) cannot be predetermined on the basis of a shallow chunk analysis (1.b). Moreover, the prevailing greedy bottom-up processing strategies applied in chunk parsing do not take into account the macro-structure of sentences. They are thus easily trapped in cases such as (2).

(1) a. $[_{\mathrm{CL}}$There was $[_{\mathrm{NP}}$ a rumor $[_{\mathrm{CL}}$ it was going to be bought by $[_{\mathrm{NP}}$ a French company $[_{\mathrm{CL}}$ that competes in supercomputers]]]]].

b. $[_{\mathrm{CL}}$There was $[_{\mathrm{NP}}$ a rumor]] $[_{\mathrm{CL}}$ it was going to be bought by $[_{\mathrm{NP}}$ a French company]] $[_{\mathrm{CL}}$ that competes in supercomputers].

(2) Fred eats $[_{\mathrm{NP}}$ pizza and Mary] drinks wine.

In sum, state-of-the-art chunk parsing does neither provide sufficient detail, nor the required accuracy to act as a 'guide' for deep syntactic analysis.

2.2 Stochastic Topological Parsing

Recently, there is revived interest in shallow analyses that determine the clausal macro-structure of sentences. The *topological field model* of (German) syntax (Höhle, 1983) divides basic clauses into distinct fields – *pre-, middle-*, and *post-fields* – delimited by verbal or sentential markers, which constitute the left/right sentence brackets. This model of clause structure is underspecified, or *partial* as to non-sentential constituent structure, but provides a theory-neutral model of sentence *macro-structure*.

Due to its linguistic underpinning, the topological field model provides a pre-partitioning of complex sentences that is (i) highly compatible with deep syntactic analysis, and thus (ii) maximally effective to increase parsing efficiency if interleaved with deep syntactic analysis; (iii) partiality regarding the constituency of non-sentential material ensures robustness, coverage, and processing efficiency.

(Becker and Frank, 2002) explored a corpus-based stochastic approach to topological field parsing, by training a non-lexicalised PCFG on a topological corpus derived from the NEGRA treebank of German. Measured on the basis of hand-corrected PoS-tagged input as provided by the NEGRA treebank, the parser achieves 100% coverage for length ≤ 40 (99.8% for all). Labelled precision and recall are around 93%. Perfect match (full tree identity) is about 80% (cf. Table 1, disamb +).

In this paper, the topological parser was provided a tagger front-end for free text processing, using the TnT tagger (Brants, 2000). The grammar was ported to the efficient LoPar parser of (Schmid, 2000). Tagging inaccuracies lead to a drop of 5.1/4.7 percent-

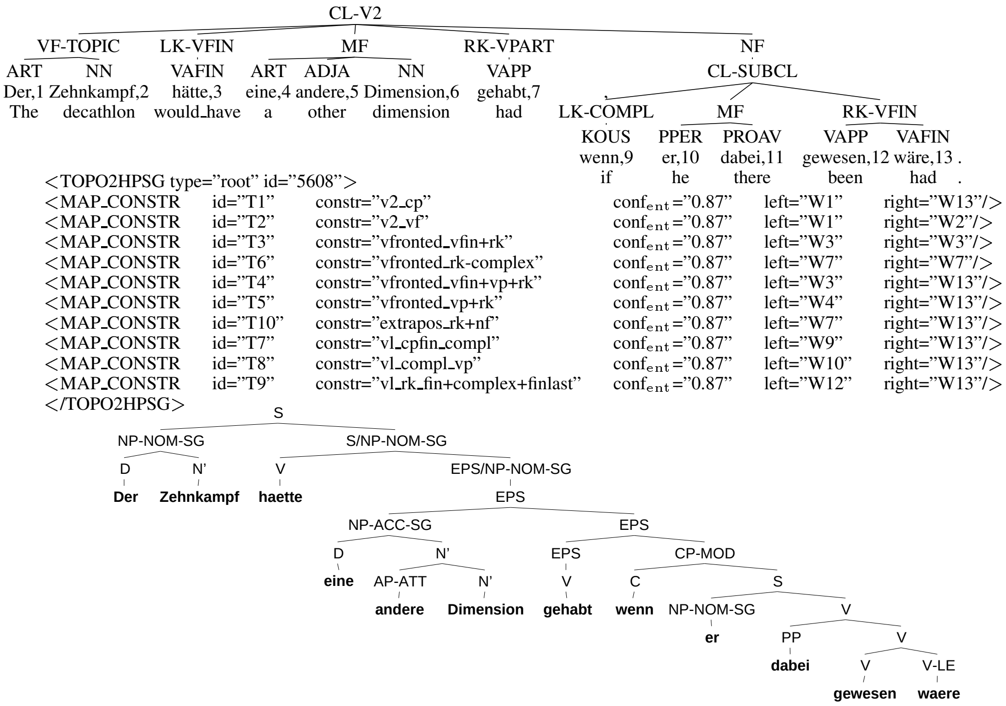

Figure 1: Topological tree w/param. cat., TOPO2HPSG map-constraints, tree skeleton of HPSG analysis

dis-amb	cove-rage	perfect match	LP in %	LR in %	0CB in %	2CB in %
+	100.0	80.4	93.4	92.9	92.1	98.9
−	99.8	72.1	88.3	88.2	87.8	97.9

Table 1: Disamb: correct (+) / tagger (−) PoS input. Eval. on atomic (vs. parameterised) category labels.

age points in LP/LR, and 8.3 percentage points in perfect match rate (Table 1, disamb −).

As seen in Figure 1, the topological trees abstract away from non-sentential constituency – phrasal fields MF (middle-field) and VF (pre-field) directly expand to PoS tags. By contrast, they perfectly render the clausal skeleton and embedding structure of complex sentences. In addition, parameterised category labels encode larger syntactic contexts, or 'constructions', such as clause type (CL-V2, -SUBCL, -REL), or inflectional patterns of verbal clusters (RK-VFIN,-VPART). These properties, along with their high accuracy rate, make them perfect candidates for tight integration with deep syntactic analysis.

Moreover, due to the combination of scrambling and discontinuous verb clusters in German syntax, a deep parser is confronted with a high degree of local ambiguity that can only be resolved at the clausal level. Highly lexicalised frameworks such as HPSG, however, do not lend themselves naturally to a top-down parsing strategy. Using topological analyses to guide the HPSG will thus provide external top-down information for bottom-up parsing.

3 TopP meets HPSG

Our work aims at integration of topological and HPSG parsing in a data-centric architecture, where each component acts independently[2] – in contrast to the combination of different syntactic formalisms within a unified parsing process.[3] Data-based integration not only favours modularity, but facilitates flexible and targeted dovetailing of structures.

3.1 Mapping Topological to HPSG Structures

While structurally similar, topological trees are not fully isomorphic to HPSG structures. In Figure 1, e.g., the span from the verb 'hätte' to the end of the sentence forms a constituent in the HPSG analysis, while in the topological tree the same span is dominated by a sequence of categories: LK, MF, RK, NF.

Yet, due to its linguistic underpinning, the topological tree can be used to systematically predict key constituents in the corresponding 'target' HPSG

[2]See Section 6 for comparison to recent work on integrated chunk-based and dependency parsing in (Daum et al., 2003).

[3]As, for example, in (Duchier and Debusmann, 2001).

analysis. We know, for example, that the span from the fronted verb (LK-VFIN) till the end of its clause CL-V2 corresponds to an HPSG phrase. Also, the first position that follows this verb, here the leftmost daughter of MF, demarcates the left edge of the traditional VP. Spans of the vorfeld VF and clause categories CL exactly match HPSG constituents. Category CL-V2 tells us that we need to reckon with a fronted verb in position of its LK daughter, here 3, while in CL-SUBCL we expect a complementiser in the position of LK, and a finite verb within the right verbal complex RK, which spans positions 12 to 13.

In order to communicate such structural constraints to the deep parser, we scan the topological tree for relevant configurations, and extract the span information for the target HPSG constituents. The resulting 'map constraints' (Fig. 1) encode a bracket type name[4] that identifies the target constituent and its left and right boundary, i.e. the concrete span in the sentence under consideration. The span is encoded by the word position index in the input, which is identical for the two parsing processes.[5]

In addition to pure constituency constraints, a skilled grammar writer will be able to associate specific HPSG grammar constraints – positive or negative – with these bracket types. These additional constraints will be globally defined, to permit fine-grained guidance of the parsing process. This and further information (cf. Section 4) is communicated to the deep parser by way of an XML interface.

3.2 Annotation-based Integration

In the annotation-based architecture of (Crysmann et al., 2002), XML-encoded analysis results of all components are stored in a multi-layer XML chart. The architecture employed in this paper improves on (Crysmann et al., 2002) by providing a central Whiteboard Annotation Transformer (WHAT) that supports flexible and powerful access to and transformation of XML annotation based on standard XSLT engines[6] (see (Schäfer, 2003) for more details on WHAT). Shallow-deep integration is thus fully annotation driven. Complex XSLT transformations are applied to the various analyses, in order to extract or combine independent knowledge sources, including XPath access to information stored in shallow annotation, complex XSLT transformations to the output of the topological parser, and extraction of bracket constraints.

3.3 Shaping the Deep Parser's Search Space

The HPSG parser is an active bidirectional chart parser which allows flexible parsing strategies by using an agenda for the parsing tasks.[7] To compute priorities for the tasks, several information sources can be consulted, e.g. the estimated quality of the participating edges or external resources like PoS tagger results. Object-oriented implementation of the priority computation facilitates exchange and, moreover, combination of different ranking strategies. Extending our current regime that uses PoS tagging for prioritisation,[8] we are now utilising phrasal constraints (brackets) from topological analysis to enhance the hand-crafted parsing heuristic employed so far.

Conditions for changing default priorities Every bracket pair br_x computed from the topological analysis comes with a bracket type x that defines its behaviour in the priority computation. Each bracket type can be associated with a set of positive and negative constraints that state a set of permissible or forbidden rules and/or feature structure configurations for the HPSG analysis.

The bracket types fall into three main categories: *left-*, *right-*, and *fully matching* brackets. A right-matching bracket may affect the priority of tasks whose resulting edge will end at the right bracket of a pair, like, for example, a task that would combine edges *C* and *F* or *C* and *D* in Fig. 2. Left-matching brackets work analogously. For fully matching brackets, only tasks that produce an edge that matches the span of the bracket pair can be affected, like, e.g., a task that combines edges *B* and *C* in Fig. 2. If, in addition, specified rule as well as feature structure constraints hold, the task is rewarded if they are positive constraints, and penalised if they are negative ones. All tasks that produce *crossing* edges, i.e. where one endpoint lies strictly inside the bracket pair and the other lies strictly outside, are penalised, e.g., a task that combines edges *A* and *B*.

This behaviour can be implemented efficiently when we assume that the computation of a task pri-

[4] We currently extract 34 different bracket types.

[5] We currently assume identical tokenisation, but could accommodate for distinct tokenisation regimes, using map tables.

[6] Advantages we see in the XSLT approach are (i) minimised programming effort in the target implementation language for XML access, (ii) reuse of transformation rules in multiple modules, (iii) fast integration of new XML-producing components.

[7] A parsing task encodes the possible combination of a passive and an active chart edge.

[8] See e.g. (Prins and van Noord, 2001) for related work.

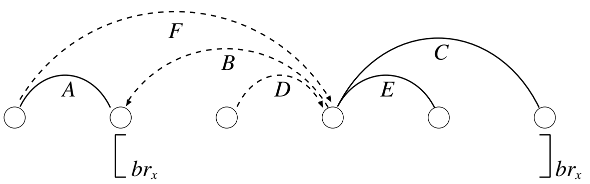

Figure 2: An example chart with a bracket pair of type x. The dashed edges are active.

ority takes into account the priorities of the tasks it builds upon. This guarantees that the effect of changing one task in the parsing process will propagate to all depending tasks without having to check the bracket conditions repeatedly.

For each task, it is sufficient to examine the start- and endpoints of the building edges to determine if its priority is affected by some bracket. Only four cases can occur:

1. The new edge spans a pair of brackets: a match
2. The new edge starts or ends at one of the brackets, but does not match: left or right hit
3. One bracket of a pair is at the joint of the building edges and a start- or endpoint lies strictly inside the brackets: a crossing (edges *A* and *B* in Fig. 2)
4. No bracket at the endpoints of both edges: use the default priority

For left-/right-matching brackets, a match behaves exactly like the corresponding left or right hit.

Computing the new priority If the priority of a task is changed, the change is computed *relative* to the default priority. We use two alternative confidence values, and a hand-coded parameter $\gamma(x)$, to adjust the impact on the default priority heuristics. $conf_{ent}(br_x)$ specifies the confidence for a concrete bracket pair br_x of type x in a given sentence, based on the tree entropy of the topological parse. $conf_{pr}$ specifies a measure of 'expected accuracy' for each bracket *type*. Sec. 4 will introduce these measures.

The priority $p(t)$ of a task t involving a bracket br_x is computed from the default priority $\tilde{p}(t)$ by:

$$p(t) = \tilde{p}(t) * (1 \pm conf_{ent}(br_x) * conf_{pr}(x) * \gamma(x))$$

4 Confidence Measures

This way of calculating priorities allows flexible parameterisation for the integration of bracket constraints. While the topological parser's accuracy is high, we need to reckon with (partially) wrong analyses that could counter the expected performance gains. An important factor is therefore the *confidence* we can have, for any new sentence, into the best parse delivered by the topological parser: If confidence is high, we want it to be fully considered for prioritisation – if it is low, we want to lower its impact, or completely ignore the proposed brackets.

We will experiment with two alternative confidence measures: (i) expected accuracy of particular bracket types extracted from the best parse delivered, and (ii) tree entropy based on the probability distribution encountered in a topological parse, as a measure of the overall accuracy of the best parse proposed – and thus the extracted brackets.[9]

4.1 Conf$_{\mathrm{pr}}$: Accuracy of map-constraints

To determine a measure of 'expected accuracy' for the map constraints, we computed precision and recall for the 34 bracket types by comparing the extracted brackets from the suite of best delivered topological parses against the brackets we extracted from the trees in the manually annotated evaluation corpus in (Becker and Frank, 2002). We obtain 88.3% precision, 87.8% recall for brackets extracted from the best topological parse, run with TnT front end. We chose precision of extracted bracket types as a static confidence weight for prioritisation.

Precision figures are distributed as follows: 26.5% of the bracket *types* have precision $\geq$ 90% (93.1% in avg, 53.5% of bracket mass), 50% have precision $\geq$ 80% (88.9% avg, 77.7% bracket mass). 20.6% have precision $\leq$ 50% (41.26% in avg, 2.7% bracket mass). For experiments using a threshold on $conf_{pr}(x)$ for bracket type x, we set a threshold value of 0.7, which excludes 32.35% of the low-confidence bracket types (and 22.1% bracket mass), and includes chunk-based brackets (see Section 5).

4.2 Conf$_{\mathrm{ent}}$: Entropy of Parse Distribution

While precision over bracket types is a static measure that is independent from the structural complexity of a particular sentence, tree entropy is defined as the entropy over the probability distribution of the set of parsed trees for a given sentence. It is a useful measure to assess how certain the parser is about the best analysis, e.g. to measure the *training utility value* of a data point in the context of sample selection (Hwa, 2000). We thus employ tree entropy as a

[9] Further measures are conceivable: We could extract brackets from some n-best topological parses, associating them with weights, using methods similar to (Carroll and Briscoe, 2002).

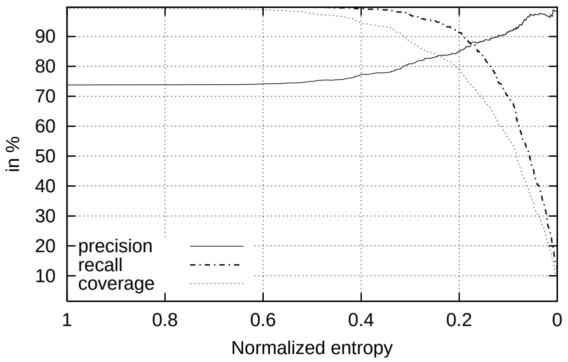

Figure 3: Effect of different thresholds of normalized entropy on precision, recall, and coverage

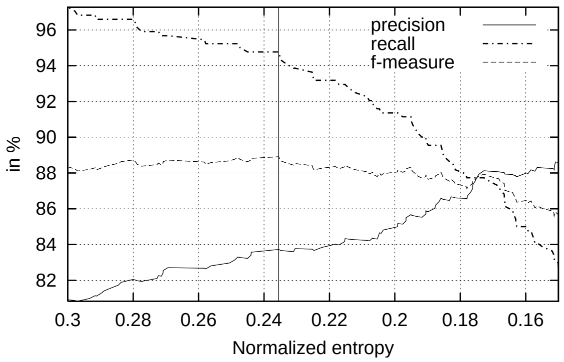

Figure 4: Maximise f-measure on the training set to determine best entropy threshold

confidence measure for the quality of the best topological parse, and the extracted bracket constraints.

We carry out an experiment to assess the effect of varying entropy thresholds θ on precision and recall of topological parsing, in terms of perfect match rate, and show a way to determine an optimal value for θ. We compute tree entropy over the full probability distribution, and normalise the values to be distributed in a range between 0 and 1. The normalisation factor is empirically determined as the highest entropy over all sentences of the training set.[10]

Experimental setup We randomly split the manually corrected evaluation corpus of (Becker and Frank, 2002) (for sentence length ≤ 40) into a training set of 600 sentences and a test set of 408 sentences. This yields the following values for the training set (test set in brackets): initial perfect match rate is 73.5% (70.0%), LP 88.8% (87.6%), and LR 88.5% (87.8%).[11] Coverage is 99.8% for both.

Evaluation measures For the task of identifying the perfect matches from a set of parses we give the following standard definitions: precision is the proportion of selected parses that have a perfect match – thus being the perfect match rate, and recall is the proportion of perfect matches that the system selected. Coverage is usually defined as the proportion of attempted analyses with at least one parse. We extend this definition to treat successful analyses with a high tree entropy as being *out of coverage*. Fig. 3 shows the effect of decreasing entropy thresholds θ on precision, recall and coverage. The unfiltered set of all sentences is found at θ=1. Lowering θ increases precision, and decreases recall and coverage. We determine f-measure as composite measure of precision and recall with equal weighting (α=0.5).

Results We use f-measure as a target function on the training set to determine a plausible θ. F-measure is maximal at θ=0.236 with 88.9%, see Figure 4. Precision and recall are 83.7% and 94.8% resp. while coverage goes down to 83.0%. Applying the same θ on the test set, we get the following results: 80.5% precision, 93.0% recall. Coverage goes down to 80.6%. LP is 93.3%, LR is 91.2%.

Confidence Measure We distribute the complement of the associated tree entropy of a parse tree tr as a global confidence measure over all brackets br extracted from that parse: $conf_{ent}(br) = 1 - ent(tr)$. For the thresholded version of $conf_{ent}(br)$, we set the threshold to $1 - \theta = 1 - 0.236 = 0.764$.

5 Experiments

Experimental Setup In the experiments we use the subset of the NEGRA corpus (5060 sents, 24.57%) that is currently parsed by the HPSG grammar.[12] Average sentence length is 8.94, ignoring punctuation; average lexical ambiguity is 3.05 entries/word. As baseline, we performed a run without topological information, yet including PoS prioritisation from tagging.[13] A series of tests explores the effects of alternative parameter settings. We further test the impact of chunk information. To this

[10]Possibly higher values in the test set will be clipped to 1.

[11]Evaluation figures for this experiment are given disregarding parameterisation (and punctuation), corresponding to the first row of figures in table 1.

[12]This test set is different from the corpus used in Section 4.

[13]In a comparative run without PoS-priorisation, we established a speed-up factor of 1.13 towards the baseline used in our experiment, with a slight increase in coverage (1%). This compares to a speed-up factor of 2.26 reported in (Daum et al., 2003), by integration of PoS guidance into a dependency parser.

end, phrasal fields determined by topological parsing were fed to the chunk parser of (Skut and Brants, 1998). Extracted NP and PP bracket constraints are defined as left-matching bracket types, to compensate for the non-embedding structure of chunks. Chunk brackets are tested in conjunction with topological brackets, and in isolation, using the labelled precision value of 71.1% in (Skut and Brants, 1998) as a uniform confidence weight.[14]

Measures For all runs we measure the absolute time and the number of parsing tasks needed to compute the first reading. The times in the individual runs were normalised according to the number of executed tasks per second. We noticed that the coverage of some integrated runs decreased by up to 1% of the 5060 test items, with a typical loss of around 0.5%. To warrant that we are not just trading coverage for speed, we derived two measures from the primary data: an upper bound, where we associated every unsuccessful parse with the time and number of tasks used when the limit of 70000 passive edges was hit, and a lower bound, where we removed the most expensive parses from each run, until we reached the same coverage. Whereas the upper bound is certainly more realistic in an application context, the lower bound gives us a worst case estimate of expectable speed-up.

Integration Parameters We explored the following range of weighting parameters for prioritisation (see Section 3.3 and Table 2).

We use two global settings for the heuristic parameter γ. Setting γ to $\frac{1}{2}$ without using any confidence measure causes the priority of every affected parsing task to be in- or decreased by half its value. Setting γ to 1 drastically increases the influence of topological information, the priority for rewarded tasks is doubled and set to zero for penalized ones.

The first two runs (rows with **−P** **−E**) ignore both confidence parameters ($conf_{pr/ent}$=1), measuring only the effect of higher or lower influence of topological information. In the remaining six runs, the impact of the confidence measures $conf_{pr/ent}$ is tested individually, namely **+P** **−E** and **−P** **+E**, by setting the resp. alternative value to 1. For two runs, we set the resp. confidence values that drop below a certain threshold to zero (**PT**, **ET**) to exclude un-

[14]The experiments were run on a 700 MHz Pentium III machine. For all runs, the maximum number of passive edges was set to the comparatively high value of 70000.

	factor		msec (1st)		tasks	
	low-b	up-b	low-b	up-b	low-b	up-b
Baseline	–	–	524	675	3813	4749
Integration of topological brackets w/ parameters						
−P −E $\gamma\frac{1}{2}$	2.21	2.17	237	310	1851	2353
−P −E $\gamma 1$	2.04	2.10	257	320	2037	2377
+P −E $\gamma\frac{1}{2}$	2.15	2.21	243	306	1877	2288
PT −E $\gamma\frac{1}{2}$	**2.20**	**2.30**	238	294	1890	2268
−P +E $\gamma\frac{1}{2}$	**2.27**	**2.23**	230	302	1811	2330
−P ET $\gamma\frac{1}{2}$	2.10	2.00	250	337	1896	2503
+P −E $\gamma 1$	2.06	2.12	255	318	2021	2360
PT −E $\gamma 1$	2.08	2.10	252	321	1941	2346
PT with chunk and topological brackets						
PT −E $\gamma\frac{1}{2}$	2.13	2.16	246	312	1929	2379
PT with chunk brackets only						
PT −E $\gamma\frac{1}{2}$	0.89	1.10	589	611	4102	4234

Table 2: Priority weight parameters and results

certain candidate brackets or bracket types. For runs including chunk bracketing constraints, we chose thresholded precision (**PT**) as confidence weights for topological and/or chunk brackets.

6 Discussion of Results

Table 2 summarises the results. A high impact on bracket constraints ($\gamma 1$) results in lower performance gains than using a moderate impact ($\gamma\frac{1}{2}$) (rows 2,4,5 vs. 3,8,9). A possible interpretation is that for high γ, wrong topological constraints and strong negative priorities can mislead the parser.

Use of confidence weights yields the best performance gains (with $\gamma\frac{1}{2}$), in particular, thresholded precision of bracket types **PT**, and tree entropy **+E**, with comparable speed-up of factor 2.2/2.3 and 2.27/2.23 (2.25 if averaged). Thresholded entropy **ET** yields slightly lower gains. This could be due to a non-optimal threshold, or the fact that – while precision differentiates bracket types in terms of their confidence, such that only a small number of brackets are weakened – tree entropy as a global measure penalizes all brackets for a sentence on an equal basis, neutralizing positive effects which – as seen in **+/−P** – may still contribute useful information.

Additional use of chunk brackets (row 10) leads to a slight decrease, probably due to lower precision of chunk brackets. Even more, isolated use of chunk information (row 11) does not yield signifi-

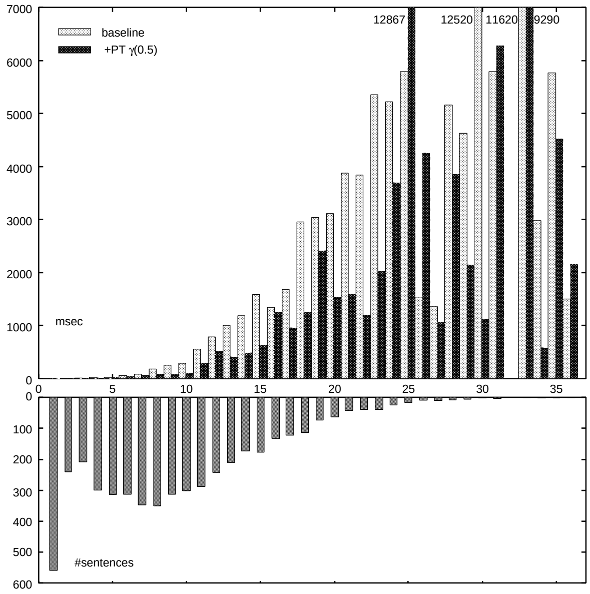

Figure 5: Performance gain/loss per sentence length

cant gains over the baseline (0.89/1.1). Similar results were reported in (Daum et al., 2003) for integration of chunk- and dependency parsing.[15]

For **PT -E** $\gamma\frac{1}{2}$, Figure 5 shows substantial performance gains, with some outliers in the range of length 25–36. 962 sentences (length >3, avg. 11.09) took longer parse time as compared to the baseline (with 5% variance margin). For coverage losses, we isolated two factors: while erroneous topological information could lead the parser astray, we also found cases where topological information prevented spurious HPSG parses to surface. This suggests that the integrated system bears the potential of cross-validation of different components.

7 Conclusion

We demonstrated that integration of shallow topological and deep HPSG processing results in significant performance gains, of factor 2.25—at a high level of deep parser efficiency. We show that macro-structural constraints derived from topological parsing improve significantly over chunk-based constraints. Fine-grained prioritisation in terms of confidence weights could further improve the results.

Our annotation-based architecture is now easily extended to address robustness issues beyond lexical matters. By extracting spans for clausal fragments from topological parses, in case of deep parsing failure the chart can be inspected for spanning analyses for sub-sentential fragments. Further, we can simplify the input sentence, by pruning adjunct subclauses, and trigger reparsing on the pruned input.

[15](Daum et al., 2003) report a gain of factor 2.76 relative to a non-PoS-guided baseline, which reduces to factor 1.21 relative to a PoS-prioritised baseline, as in our scenario.

References

M. Becker and A. Frank. 2002. A Stochastic Topological Parser of German. In *Proceedings of COLING 2002*, pages 71–77, Taipei, Taiwan.

T. Brants. 2000. Tnt - A Statistical Part-of-Speech Tagger. In *Proceedings of Eurospeech*, Rhodes, Greece.

U. Callmeier. 2000. PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6 (1):99 – 108.

C. Carroll and E. Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of COLING 2002*, pages 134–140.

B. Crysmann, A. Frank, B. Kiefer, St. Müller, G. Neumann, J. Piskorski, U. Schäfer, M. Siegel, H. Uszkoreit, F. Xu, M. Becker, and H.-U. Krieger. 2002. An Integrated Architecture for Deep and Shallow Processing. In *Proceedings of ACL 2002*, Pittsburgh.

M. Daum, K.A. Foth, and W. Menzel. 2003. Constraint Based Integration of Deep and Shallow Parsing Techniques. In *Proceedings of EACL 2003*, Budapest.

D. Duchier and R. Debusmann. 2001. Topological Dependency Trees: A Constraint-based Account of Linear Precedence. In *Proceedings of ACL 2001*.

C. Grover and A. Lascarides. 2001. XML-based data preparation for robust deep parsing. In *Proceedings of ACL/EACL 2001*, pages 252–259, Toulouse, France.

T. Höhle. 1983. Topologische Felder. Unpublished manuscript, University of Cologne.

R. Hwa. 2000. Sample selection for statistical grammar induction. In *Proceedings of EMNLP/VLC-2000*, pages 45–52, Hong Kong.

S. Müller and W. Kasper. 2000. HPSG analysis of German. In W. Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, Artificial Intelligence, pages 238–253. Springer, Berlin.

R. Prins and G. van Noord. 2001. Unsupervised pos-tagging improves parsing accuracy and parsing efficiency. In *Proceedings of IWPT*, Beijing.

U. Schäfer. 2003. WHAT: An XSLT-based Infrastructure for the Integration of Natural Language Processing Components. In *Proceedings of the SEALTS Workshop, HLT-NAACL03*, Edmonton, Canada.

H. Schmid, 2000. *LoPar: Design and Implementation*. IMS, Stuttgart. Arbeitspapiere des SFB 340, Nr. 149.

W. Skut and T. Brants. 1998. Chunk tagger: statistical recognition of noun phrases. In *ESSLLI-1998 Workshop on Automated Acquisition of Syntax and Parsing*.

H. Uszkoreit. 2002. New Chances for Deep Linguistic Processing. In *Proceedings of COLING 2002*, pages xiv–xxvii, Taipei, Taiwan.