Large-Scale Corpus-Driven PCFG Approximation of an HPSG

Yi Zhang
LT-Lab, DFKI GmbH
Saarbrücken, Germany
yizhang@dfki.de

Hans-Ulrich Krieger
LT-Lab, DFKI GmbH
Saarbrücken, Germany
krieger@dfki.de

Abstract

We present a novel corpus-driven approach towards grammar approximation for a linguistically deep Head-driven Phrase Structure Grammar. With an unlexicalized probabilistic context-free grammar obtained by Maximum Likelihood Estimate on a large-scale automatically annotated corpus, we are able to achieve parsing accuracy higher than the original HPSG-based model. Different ways of enriching the annotations carried by the approximating PCFG are proposed and compared. Comparison to the state-of-the-art latent-variable PCFG shows that our approach is more suitable for the grammar approximation task where training data can be acquired automatically. The best approximating PCFG achieved ParsEval F_1 accuracy of 84.13%. The high robustness of the PCFG suggests it is a viable way of achieving full coverage parsing with the hand-written deep linguistic grammars.

1 Introduction

Deep linguistic processing technologies have been evolving closely around the development of rich formalisms which typically introduce mild context-sensitivity. Examples of well-adopted frameworks include various Tree Adjoining Grammars, Combinatory Categorial Grammars, Lexical Functional Grammars (with untyped Features Structures in F-structures), and Head-Driven Phrase Structure Grammars (with Typed Featured Structures). Such formalisms have been successfully powering the modern formal linguistic studies. However, the intrinsic complexity of *deeper formalisms*[1] hinders the deployment of such resources in language technology applications.

Take HPSG for example. The linguistic framework is built on top of the typed feature logic formalisms (e.g., Carpenter (1992)). The monostratal representation integrates various syntactic and semantic information concerning a linguistic object (and all its sub-components) in a single typed features structure. And the integration of information and compatibility checking is achieved by the *unification* operation. Such a formalism is especially suitable for developing and implementing a linguistic theory. But the lack of a polynomial upper-bound time complexity in unification-based parsing raises concerns of the processing efficiency.

Meanwhile, from the grammar engineering perspective we see grammar developers constantly joggling between two somewhat conflicting goals: on the one hand, to describe the linguistic phenomena in a precisely constrained way; on the other hand, to achieve broad coverage when parsing unseen real-world texts. As a result, many of these large-scale grammar implementations are forced to choose to either compromise the linguistic preciseness, or to accept the low coverage in parsing.

In this paper, we propose PCFG approximation as a way to alleviate some of these issues in HPSG processing. While HPSG framework is great for linguistic description, we show that when carefully designed, a much simpler approximating probabilistic context-free grammar can be extracted automatically, and is capable of achieving good parsing accuracy while maintaining high robustness and efficiency.

The rest of the paper is organized as the following: Section 2 gives an overview of HPSG as an linguistic theory and its application in parsing;

[1]In the context of this paper, by deeper formalism we mean formalisms which are at least mildly context-sensitive, in a comparative sense to the context-free grammars.

Section 3 reviews the previous related work on CFG approximation of HPSG; Section 4 presents the corpus-driven PCFG approximation with both internal and external annotations; Section 5 describes the evaluation setup and results; Section 6 compares our approach with other related work on parsing (such as self-training), and foresees the application of the approximating PCFGs; Section 7 concludes the paper.

2 Parsing with Head-Driven Phrase Structure Grammars

Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994) is a constraint-based highly lexicalized non-derivational generative grammar framework. Based on a typed feature structures formalism, the HPSG theory describes a small set of highly generalized linguistic principles, with which the rich information derived from the detailed lexical types interacts to produce precise linguistic interpretations.

Several large-scale HPSG-based NLP parsing systems have been built in the past decade. Among them are the Enju English & Chinese parser (Miyao et al., 2004; Yu et al., 2010), the Alpino parser for Dutch (van Noord, 2006), and the LKB & PET (Copestake, 2002; Callmeier, 2000) for English, German, Japanese and a dozen more other DELPH-IN[2] grammars of various languages. These systems are successful showcases of where modern grammar engineering contributes to the state-of-the-art parsing systems. With the modern processing techniques such as quasi-destructive unification (Tomabechi, 1991), quick check (Kiefer et al., 1999), ambiguity packing (Oepen and Carroll, 2000) and selective unpacking (Zhang et al., 2007), the practical parsing efficiency has been greatly improved. But none of these changes the underlying formalism, therefore the parser still can run into exponential parsing time in theory.

Another disadvantage of deep grammar lies in the difficulty of proper statistical modeling of the richer representation. For example, Abney (1997) shows that naïve MLE is not consistent for unification-based grammars, and proposes random fields as an alternative. In practice, we see most HPSG parsing systems opt for a discriminative *Maximum Entropy Model* (MEM) for parse ranking on top of the hypothesis space licensed by the HPSG grammar (Miyao and Tsujii, 2002; Malouf and van Noord, 2004; Toutanova et al., 2005). For further efficiency, the hypothesis space of the HPSG parses is pruned with supertagging or symbolic CFG filtering rules (Matsuzaki et al., 2007) at early stages. It is however unclear how these separate models can be unified to guide the best-first construction of HPSG parses without an exhaustive creation of the (packed) parse forest or ad hoc pruning.

Moreover, the heavily constraint-based nature of the grammar poses a difficult choice between linguistic preciseness and practical parsing robustness. As a result, many HPSG parser implementations have to sacrifice on the linguistic side in trade for a decent parsing coverage.

3 Related Work

Previous work in the direction of HPSG approximation has seen two major approaches: grammar-based approach and the corpus-driven approach.

The grammar-based approach (Kiefer and Krieger, 2004) tries to compile out a huge set of categories by flattening the TFSes into atomic symbols. This approach can in theory guarantee the equivalence of the grammars in both parsing and generation. However, in practice it generates billions of CFG productions. Even when carefully choosing a subset of the HPSG features, the resulting grammar is too large to be useful for parsing or generation.

The corpus-based approach (Krieger, 2007), on the other hand, builds the approximating CFG by observing the growth of the chart when parsing texts with the HPSG. Passive edges on the chart represent the successful application of HPSG rules, hence are modeled by an approximating CFG production. Also, the symbols in CFG only carry partial information from a small set of feature-paths used in *quick check* (Kiefer et al., 1999), i.e., the frequently failed feature-paths in unification, hence the most discriminating ones.

Both approaches are symbolic in the sense that there is no probabilistic model produced to disambiguate the CFG parses. In the case of corpus-based approach, one can also acquire frequency counts for each CFG rule. But since not all passive edges occur in a full parse tree, and not all parses are correct, the statistics obtained is not suitable for the parsing task.

Kiefer et al. (2002) propose to use a PCFG in a

[2]http://www.delph-in.net/

two-stage parsing setup, where the PCFG predicts derivations in the first step, followed by the replay of unification. The experiment was carried out on a relatively small grammar. And due to the unavailability of large-scale treebank at the time, unsupervised Inside-Outside algorithm was used for the probability estimation.

Cahill et al. (2004) reported an application of the PCFG approximation technique in LFG parsing and the recovery of long distance dependencies on the f-structures. Two main differences between their work and the one presented in this paper should be noted. First, the approximation target for Cahill et al. (2004) is a treebank-induced grammar, while this paper targets for a large-scale hand-crafted grammar. Second, the monostratal representation in HPSG entails that a correct derivation tree will also guarantee the correct recovery of unbounded dependencies represented by the underlying feature structures, while in the LFG universe these have to be resolved on the f-structures instead of the c-structures.

4 Probabilistic Context-Free Approximation of HPSG

Unlike the approaches in previous work which approximates the symbolic HPSG alone, we propose a PCFG approach which approximates the combination of the grammar and its disambiguation model. This allows us to closely model the deep parser behavior with a single approximation PCFG.

For the experiments in this paper, we use the English Resource Grammar (ERG; Flickinger (2002)) and the accompanying treebanks (see Section 5.1 for detailed descriptions). But the technique presented in this section can be easily applied to other languages and HPSG grammar implementations.

4.1 Derivation Normalization

A complete HPSG analysis is recorded in a derivation tree. The terminals of the tree are surface tokens in the sentence. The preterminals are the names of the activated lexical entries. The non-(pre)terminal nodes (except for the root) correspond to grammar rules applied to create the HPSG signs. An extra root node denotes the *"root"* condition fulfilled to license a complete HPSG parse. An example derivation of ERG is given in Figure 1.

Before extracting the approximation grammar, we perform several normalizing transformations on the original derivations. First, in order to acquire an unlexicalized grammar, we replace the lexical entry names on the preterminal with their corresponding lexical type defined in the ERG lexicon. Second, we collapse the unary chain of morphological rules together with the preterminal lexical types to form the so-called *"supertag"*. As shown in Figure 1, these unary rules always occur above the preterminals and below any syntactic constructional rules. Practice shows that this helps improve the parsing accuracy of the PCFG. The last normalization concerns with the treatment of punctuations. In ERG (as for release 1010), punctuations are treated as affixes instead of independent tokens by themselves. For better compatibility with other annotations, we convert the original punctuation-attaching unary rule (applied above the morphological rules and below the syntactic rules) into a binary branch. The normalized derivation tree of the previous example is shown in Figure 2. It is worth noting that all the normalizing steps can be reversed without introducing ambiguity. The approximating PCFGs will be extracted from the normalized derivations, while the evaluation will be reported on the original derivations (though the tagging accuracy will be calculated on the lexical types).

4.2 PCFG with External Annotation

Although the derivation tree records all necessary information to recover the complete HPSG analysis (i.e. carrying out unification on each node of the tree with corresponding grammar rules and lexical entries), it does not always encode the necessary information in an explicit way, due to the fact that rules in HPSG are highly generalized (see Section 2). For example, the rule *"hd-cmp_u_c"* in ERG can be used to express a head-complement composition without specifying the syntactic category of the head. Thus a node marked only with *"hd-cmp_u_c"* could be a VP, PP, or $\overline{\text{NP}}$. Therefore it will be difficult to accurately predict the derivation without such information. To compensate for the lack of information in the derivation tree, we add additional *annotations* to the non-terminals. We further differentiate external annotation, i.e., additional information from the context of the tree node, and internal annotation, i.e. infor-

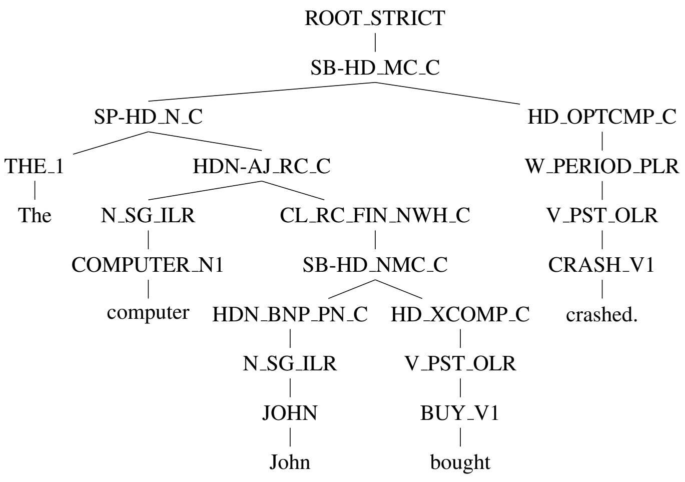

Figure 1: Example of an original ERG derivation tree

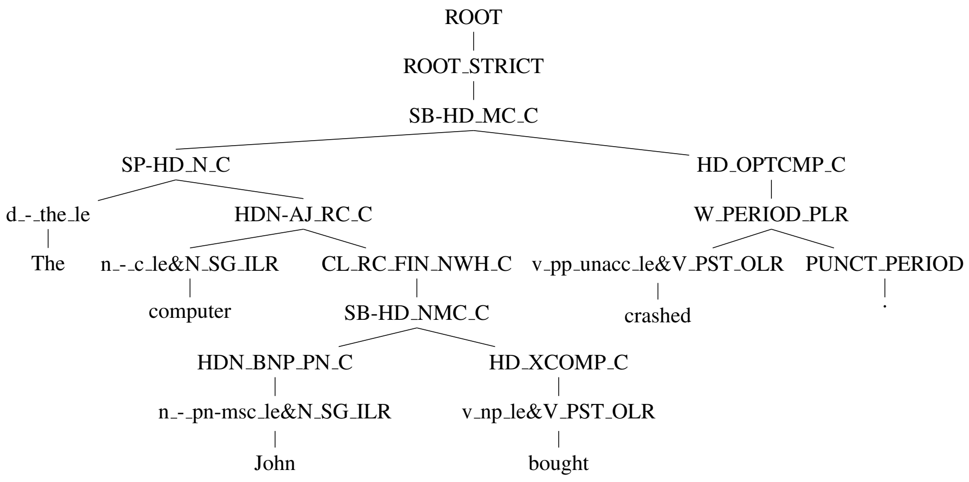

Figure 2: Example of a normalized derivation tree

mation coming from the HPSG sign.[3]

For the external annotation, we mark each nonterminal node with up to n grandparents. This is an effective technique used in both PCFG parsing (Klein and Manning, 2003)[4] and HPSG parse disambiguation (Toutanova et al., 2005). As ERG rules are either unary or binary, we do not annotate nodes with sibling information, though for a grammar with flat rules this could potentially help, as shown by Klein and Manning (2003) (so-called *horizontal markovization*). We choose not to annotate the preterminal supertags with grandparents, for the overly fine-grained tagset hurts the parsing coverage.

4.3 PCFG with Internal Annotation

While the external annotation enrich the derivation tree by gathering context information, the internal annotation explores the detailed HPSG sign associated with the tree node. Note that an average ERG sign contains hundreds of feature-paths and their corresponding values, it is important to pick a suitable small yet effective subset of them as annotation. Following the practice of (Krieger, 2007), we choose to use up to six top-ranked *quick-check* paths (see Table 1), which are the most frequently failed feature-path in unification.

To access the HPSG sign, feature structures are reconstructed by unifying the corresponding TFS of the HPSG rule with the instantiated TFSes of its daughters. This can be done efficiently even with naïve unification algorithms, for there is no search involved. And the unification never fails when the original derivation tree is produced by the ERG. Next, the value of the annotation is determined by the *type* of the TFS at the end of each given feature-path (or **undef** in case the path was not defined in the TFS). For example, for feature-path SYNSEM.LOCAL.CAT.COMPS (the remaining list of complements for the sign), value **null** denotes an empty list, while **olist** denotes a list with only optional complements. Figure 3 shows an example of an annotated tree with 1-level grandparent and HEAD feature-path annotation.

[3]Our notion of *internal* and *external* annotation is slightly different to that of (Klein and Manning, 2003). In our notion, *internal* annotation refers to the information from the local HPSG sign.

[4]This technique is called *vertical markovization* in (Klein and Manning, 2003).

	Feature-Path
1	SYNSEM.LOCAL.CAT.HEAD
2	SYNSEM.LOCAL.CONJ
3	SYNSEM.LOCAL.AGR.PNG.PN
4	SYNSEM.LOCAL.CAT.VAL.COMPS
5	SYNSEM.LOCAL.CAT.HEAD.MOD
6	SYNSEM.LOCAL.CAT.VAL.COMPS.FIRST.OPT

Table 1: Top feature-paths used for internal annotation

4.4 Grammar Extraction & Probability Estimation

To extract the approximating PCFG, we need a disambiguated treebank annotated with HPSG derivations. The treebank is constructed by first parsing the input sentences with the HPSG parser, then disambiguated either manually or automatically by the parse selection model. The CFG symbols and production rules are extracted directly from the annotation enriched derivation trees from the treebank. Each tree node contributes to one frequency count of the corresponding CFG rule with the parent's symbol as the LHS, and the symbols of its daughters as the RHS. For the experiments in this paper, we do not prune the CFG symbols or rules. The rule probability is calculated with *Maximum Likelihood Estimate* (MLE) without smoothing.

$$P_r(A \rightarrow \beta) = P(A \rightarrow \beta | A) = \frac{\#(A \rightarrow \beta)}{\#A} \quad (1)$$

The lexical model, however, does receive smoothing for unknown word handling. More specifically, words are assigned a signature ($sig(w)$) based on their capitalization, suffix, digit and other character features. We then use the MLE estimate of $P(T|sig(w))$ as a prior against which observed taggings T were taken:

$$P(T|w) = \frac{\#(T,w) + \alpha \cdot P(T|sig(w))}{\#T + \alpha} \quad (2)$$

$P(T|w)$ is then inverted to give $P(w|T)$.

The grammar extraction procedure is very efficient. The time required is linear to the size of the treebank. Even with the richest annotations (with unification operations involved), the procedure marches through thousands of trees per minute. This allows us to scale up the extraction with millions of trees.

4.5 Hierarchically Split-Merge PCFG

For comparison we also trained a hierarchically split-merge latent-variable PCFG with the Berkeley parser (Petrov et al., 2006). The latent-variable

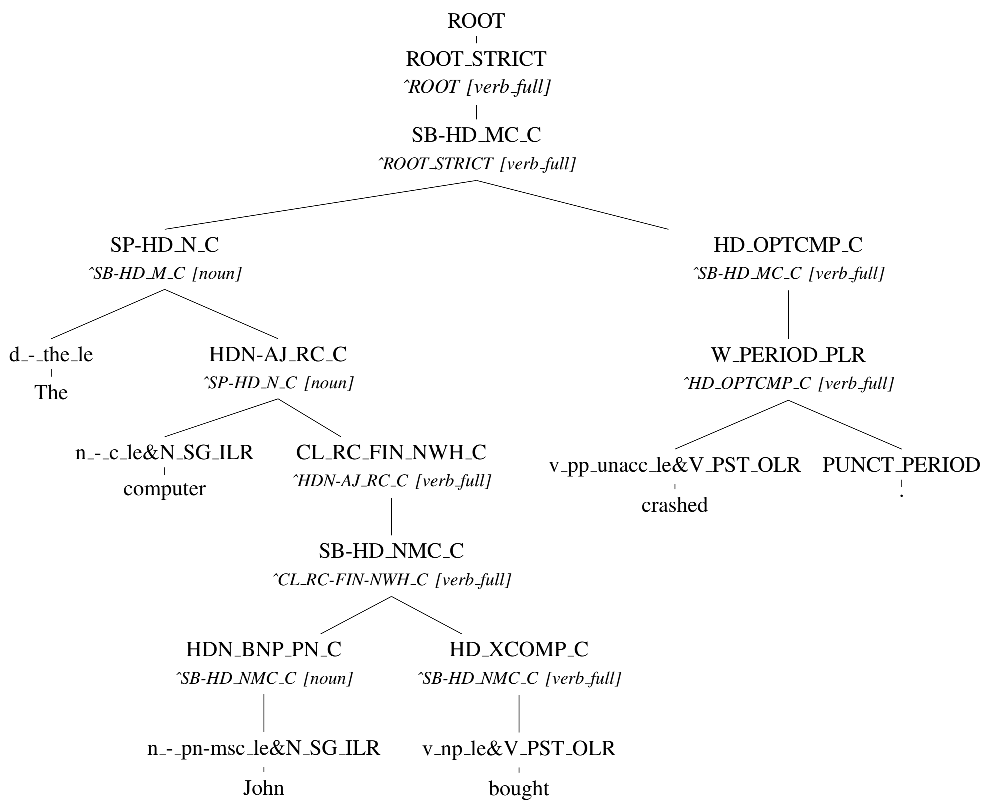

Figure 3: Example tree with 1-level grandparent and HEAD feature-path annotation

approach has proven to deliver state-of-the-art parsing performance for multiple languages. The key advantage is that it automatically induces subcategories from the treebank and produces a finer grained grammar without manual intervention. On treebanks with coarse-grained categories (which is typical for manually annotated treebanks), this is particularly effective.

In our experiment, we train the split-merge latent-variable PCFG on the derivation trees. The *Expectation-Maximization* training process is much more expensive than our MLE-based PCFG extraction. Also, given that the categories in the normalized derivations are already quite fine-grained (hundreds of non-terminal symbols and thousands of tags), the grammar stopped improving after only three rounds of split-merge iterations.

5 Experiments

5.1 Grammar & Data

We use the 1010 release of the *English Resource Grammar* for the approximation experiments. This version of the ERG contains a total of 200 syntactic constructional rules, and 50 lexical/inflectional rules. 145 of the 200 syntactic rules are binary, while the remaining 55 are unary. All lexical/inflectional rules are unary. In addition, the grammar contains a hand-compiled lexicon with around 1000 leaf lexical types and over 35K base-form entries.

Several large treebanks have been developed with the ERG. Unlike the traditional manually annotated treebanks, these are referred to as the Redwoods-style dynamic treebanks (Oepen et al., 2002). Sentences from the corpus are first parsed by the ERG, and then manually disambiguated (mostly by the grammarian himself). For the experiments in this paper, we use the manually disambiguated WeScience Treebank (Ytrestoel et al., 2009), which currently contains a totally of over 11K sentences from a selection of Wikipedia articles in the domain of Natural Language Processing with an average length of 18 tokens per sentence, pre-processed to strip irrelevant markups, and divided into 13 sections. Of all the sentences, about 78% received exactly one “gold” analysis. The rest either fail to be parsed by the ERG, or there is no single acceptable reading. We will only use the subset of sentences with a *“gold”* parse for the experiment. More specifically we keep *"ws12"* for development and *"ws13"* for the final testing. Sections *"ws01"* to *"ws11"* contain a total of 7,636 "gold" trees, which are used for training.

Apart from the WeScience, we also use the large-scale automatically disambiguated WikiWoods Treebank (Flickinger et al., 2010). Currently, WikiWoods contains about 55M English sentences extracted from the English Wikipedia articles. The corpus is parsed with the 1010 version of the ERG, and automatically disambiguated with a Maximum Entropy model trained with the manually disambiguated trees. Only 1 top-ranked reading is preserved. Since the correctness of the parse is unchecked, the dataset is potentially noisy. The total amount of trees available for training is about 48M.

5.2 The Parser

The approximating PCFG tends to grow huge when rich annotations and large corpora are used. For efficient application of the resulting grammar, we implemented a CKY-style parser with bit-vector-based algorithm as the one proposed by Schmid (2004). The algorithm shows its strength in extensibility for grammars with millions of rules and hundreds of thousands of non-terminal symbols.

A slight deviation of our implementation from the BitPar algorithm is that, after constructing the bit-vector-based recognition chart, we do not apply the top-down filtering routine before calculating the Viterbi probabilities. Practice shows that in our case the recognition chart is normally sparse, and the filtering routine itself costs more time than what it saves from the additional calculations in the Viterbi step.

For correctness checking, we reproduced the unlexicalized PCFG parsing accuracy reported by Klein and Manning (2003) on PTB with our bit-vector parser while achieving better efficiency (in both training and testing) than the Stanford Parser. Even though our parser is implemented in Java (for better cross-platform compatibility), the low-level bit-vector-based operations make our system competitive even in comparison to the BitPar implemented in C++.

As mentioned early, we do not prune the PCFG rules during parsing. For the lexical look-up, we allow the lexical model to propose multiple tags (cut by certain probability threshold). In case a full parse is not found, a partial parsing model tries to recover fragmented analysis according to the Viterbi probabilities of the constituents. With careful design of the PCFG and sufficient training data, the parser normally delivers close to full parsing coverage even without the fragmented partial parsing mode.

5.3 Evaluation & Results

For the evaluation of our approximating PCFGs, we compare the top-1 parses produced by the PCFG with the manually disambiguated gold trees in *"ws13"*. We assume the inputs have been pre-tokenized but not tagged. All comparisons are done on the original derivations. Several accuracy measures are used, including the ParsEval labeled bracketing precision, recall, F_1 and exact match ratio. Since the ParsEval scores ignore the preterminal nodes, we also report the (lexical type) tagging accuracy. Several different training sets are used.

WS contains 7636 *"gold"* trees from the sections *"ws01-ws11"* of the WeScience. The MEM parse selection model is trained with this dataset. The dataset is too small to acquire high coverage PCFGs with heavy annotations. Therefore, only PCFG(0) and PCFG(FP1) results are reported here.

WW000 contains 85,553 automatically parsed and disambiguated trees from the WikiWoods (all fragments with 000 as suffix). This is less than 0.2% of the entire WikiWoods, but close to the limit for the latent-variable PCFG training with the Berkeley Parser.

WW00 contains about 482K sentences (all fragments with 00 as suffix), roughly 1% of the entire WikiWoods. We were able to successfully train PCFGs with relatively rich annotations.

WW contains the complete WikiWoods with ~48M parsed tress. With feature-path annotations, the training of the models takes too long. Also, excessive annotation makes it difficult to parse with the resulting huge grammar. We stop at two levels of grandparent annotation, a PCFG with almost 4M rules and over 128K non-terminal symbols.

Table 2 summarizes the results of the accuracy evaluation. All results are reported on the 785 trees from the section *"ws13"* of the WeScience. MEM is the accuracy of the HPSG parser disambiguation model given the candidate parse forest. PCFG(0) is the unannotated PCFG read off the bare (normalized) derivation trees. PCFG(GPm,FPn) is the annotated PCFG model with m-levels of grandparents and n feature-paths. And PCFG-LA(SM3) is the latent-variable PCFG after three split-merge iterations.

With the small WS training set, the baseline PCFG without annotation achieved F_1 of merely 60.96. Even one level of grandparent annotation, the parsing coverage drops by over 10%. With 1 feature-path annotation, F_1 improved significantly to 66.45.

On the larger WW-000, the performance of the baseline PCFG decreases by two 2% of F_1, most likely due to the noise introduced by the automatic disambiguation. However, the larger training set enables 1-level grandparent annotation, which brings F_1 up to 71.42. The latent-variable PCFG also performs well, delivering the best F_1 at 71.87 after three split-merge iterations. But the learning curve of the Berkeley parser has already flatten out at this point, and we were unsuccessful in further scaling up the training set.

With our annotated PCFGs, significant improvements are achieved on the even larger WW-00. The baseline PCFG seems to have recovered from the previous drop, and outperforms the one trained with the *"gold"* trees. With the mixture of 1-level of grandparent and some feature-path annotations, F_1 reached over 80. The best performance on WW-00 is achieved with PCFG(GP1,FP5). 2-levels of grandparents alone outperforms 1-level of grandparent, but the grammar quickly reaches its size limit on this training set and starts to loose coverage when more feature-path annotations are added.

Finally, with the complete WikiWoods, both PCFG(GP1) and PCFG(GP2) improved further, with PCFG(GP2) reaching the highest F_1 at 84.13. It is interesting to note that this is even higher than the F_1 of the MEM disambiguation model. This is partially due to the self-training effect on the huge corpus. Another explanation is that the objective function of the discriminative MEM was optimized on the complete parse match instead of individual constituents, which will explain its high exact match ratio at 43.57%.

6 Discussion

It is important to note that the grammar approximation task we take on in this paper is different from the traditional treebank-based parsing. Although the accuracy evaluation for both tasks are done on a fixed set of *"gold"* trees, in the grammar approximation task we have access to the theoretically infinite pool of training data automatically generated by the original grammar. Some complex grammar extraction algorithms which work fine on a small training corpus fail to scale up to handle millions of trees. On the other hand, our MLE-based PCFG extraction shows its advantage in extensibility.

The approach of training a PCFG with automatically annotated treebank is in a sense similar to the self-training approach in semi-supervised parsing (McClosky et al., 2006). However, instead of parsing the unlabeled data with the PCFG directly, we rely on the HPSG grammar and its disambiguation model. The highly constrained ERG analyses on unseen data allow us to obtain high quality trees. And the penalty on introducing noisy data is quickly compensated by the huge amount of data.

The approximating PCFG is much less constrained than the ERG. From the linguistic point of view, it is difficult to interpret the huge set of PCFG rules. And unlike the ERG, the PCFG is unsuited in making grammaticality judgment. However, for the parsing task, the approximating PCFG has its advantage in being flexible and robust, needless to mention its cubic parsing time complexity. One can choose various combinations of annotations for a balanced efficiency, accuracy and coverage. Although the experiments reported in Section 5 are only testing on sentences which the ERG can parse, we have also applied the PCFGs on the remaining $\sim$20% texts and got close to full parsing coverage (less than 1% failure). Although the derivation tree constructed by the PCFG does not guaratee a unifiable HPSG analysis with typed feature structures, it provides an approximate prediction on how the HPSG analysis should look like. It is conceivable that with robust unification under the open-world assumption of the type hierarchy (Fouvry, 2003), one can get a partially well-formed semantic structure with the guidance of the approximating PCFG. Also, it would be interesting to evaluate its impact on the overall parser performance based on the semantic structures instead of the theory-specific derivations.

		#Rule	#NT	#T	P	R	F_1	Ex	TA
ws	MEM	-	-	-	82.70	82.91	**82.80**	43.57	96.87
	PCFG(0)	10,251	208	1,152	63.53	58.59	60.96	19.49	86.19
	PCFG(FP1)	12,178	669	1,152	70.02	63.22	66.45	20.51	86.20
ww−000	PCFG(0)	25,859	236	1,799	60.18	57.34	58.73	14.14	85.33
	PCFG(GP1)	64,043	3,983	1,799	72.61	70.28	71.42	21.02	86.92
	PCFG-LA(SM3)	*	*	*	73.12	70.66	**71.87**	23.18	89.81
ww−00	PCFG(0)	61,426	247	2,546	63.61	61.14	62.35	16.56	88.59
	PCFG(GP1)	187,852	5,828	2,546	77.87	77.41	77.64	24.84	91.83
	PCFG(GP1,FP4)	271,956	16,731	2,546	80.60	79.84	80.22	29.04	93.20
	PCFG(GP1,FP5)	319,511	21,414	2,546	80.94	80.32	**80.63**	28.54	93.33
	PCFG(GP1,FP6)	320,630	21,694	2,546	80.92	80.31	80.61	28.41	93.33
	PCFG(GP2)	489,890	45,658	2,546	79.68	79.56	79.62	28.92	92.01
	PCFG(GP2,FP2)	559,006	66,218	2,546	79.78	79.46	79.62	32.23	92.71
ww	PCFG(GP1)	1,007,563	8,852	4,472	80.34	79.60	79.97	28.79	93.45
	PCFG(GP2)	3,952,821	128,822	4,472	84.27	83.98	**84.13**	37.71	94.39

Table 2: Parsing Accuracy on *'ws13'* with various models and training sets. Reported are grammar size (#Rule, #NT, #T); ParsEval precision (P), recall (R), F_1, and exact match ratio (Ex) on the original derivation tree; and tagging accuracy (TA) on the lexical types.

Looking at the specific annotation strategies, we compared the internal annotations with the external ones. Experiment result shows that when the grandparent annotation is added, the grammar size grows quickly. On a huge training set, this is rewarded with significant accuracy gain. On the smaller training set though, over-annotating with grandparents results in a decrease in accuracy due to data sparseness. Instead, annotating with feature-path information increases the grammar size moderately, allowing one to approach the optimal granularity of the PCFG.

In comparison to the linguistic annotations used by Klein and Manning (2003) for PTB parsing, our annotations are less language or treebank specific. This is due to the fact that the ERG rules are relatively fine-grained in treating various linguistic constructions. And the most relevant annotations can be gathered from either the grandparents or the internal feature structure. Such general design allows us to experiment with other deep HPSG grammars in the near future.

For the clarity of the experiment, we have chosen not to do constructional pruning or smoothing, and focused our evaluation mostly on parsing accuracy. This leaves much room for future investigation. For instance, we observe that a large portion of the grammar rules have very low frequency counts and almost no impact on the parsing accuracy. On the other hand, even with the simple PCFG(GP1), after training with 45M sentences, the grammar continues to pick up new rules at a rate of one rule per 160 sentences. Most of these new rules are the combination of low frequency supertags.

Last but not the least, given the promising parsing accuracy of the approximating PCFG, we believe it is worth reconsidering the role of the hand-written grammars in the deep linguistic processing. In the past, manual grammar engineering has been taking on the dual role of offering concise and accurate linguistic description on the one hand, while attending the efficiency and robustness in parsing on the other. The conflicting goal has hindered the development of large-scale linguistic grammars. The technique as the one presented in this paper shows one way of liberating grammarians from the concerns over the processing difficulties.

7 Conclusion

We presented a corpus-driven approach to approximate a large-scale hand-written HPSG with a PCFG for robust and accurate parsing. Different annotations are used to enrich the derivation trees. And with the 48M sentence from the English Wikipedia automatically parsed and disambiguated by the ERG, a MLE-based PCFG achieved F_1 of 84.13, higher than the performance of the discriminative MEM parse selection model (which has access to the candidate HPSG parse forest). The obvious robustness and potential efficiency advantages of the approximating PCFG suggest its promising applications in deep linguistic processing.

Acknowledgments

We thank Stepehn Oepen, Bernd Kiefer and Dan Flickinger for enlightening discussions related to the work presented in this paper. We also thank the anonymous reviewers for their comments. The first author thanks the DFG-funded *Excellence Cluster of MMCI* and the *Deependance* project funded by the BMBF for their support of the work. The second author thanks project CogX (FP7 ICT 215181), NIFTi (FP7 ICT 247870) and Monnet (FP7 ICT 248458) for their support.

References

Steven Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23:597–618.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 319–326, Barcelona, Spain.

Ulrich Callmeier. 2000. PET – a platform for experimentation with efficient HPSG processing techniques. *Journal of Natural Language Engineering*, 6(1):99–108.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, UK.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI, Stanford, USA.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestl. 2010. WikiWoods: Syntacto-semantic annotation for English Wikipedia. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications.

Frederik Fouvry. 2003. *Robust processing for constraint-based grammar formalisms*. Ph.D. thesis, Gradudate School, University of Essex, Colchester, UK.

Bernd Kiefer and Hans-Ulrich Krieger. 2004. A context-free superset approximation of unification-based grammars. In *New Developments in Parsing Technology*. Kluwer. http://www.wkap.nl/prod/b/1-4020-2293-X.

Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A Bag of Useful Techniques for Efficient and Robust Parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, Maryland, USA.

Bernd Kiefer, Hans-Ulrich Krieger, and Detlef Prescher. 2002. A novel disambiguation method for unification-based grammars using probabilistic context-free approximations. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, Taipei, Taiwan.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Hans-Ulrich Krieger. 2007. From UBGs to CFGs: A practical corpus-driven approach. *Natural Language Engineering*, 13(4):317–351. Published online in April 2006.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP-04 Workshop: Beyond Shallow Analyses - Formalisms and Statistical Modeling for Deep Analyses*, Hainan Island, China.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the 20th International Joint Conference* *on Artificial Intelligence (IJCAI 2007)*, pages 1671–1676, Hyderabad, India.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL-2006*, pages 152–159, New York, USA.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference*, San Diego, USA.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP 2004)*, pages 684–693, Hainan Island, China.

Stephan Oepen and John Carroll. 2000. Ambiguity packing in constraint-based parsing — practical results. In *Proceedings of the 1st Annual Meeting of the North American Chapter of Association for Computational Linguistics (NAACL 2000)*, pages 162–169, Seattle, USA.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: motivation and preliminary applications. In *Proceedings of COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, Taipei, Taiwan.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia.

Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of Coling 2004*, pages 162–168, Geneva, Switzerland.

Hideto Tomabechi. 1991. Quasi-destructive graph unification. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL 1991)*, pages 315–322, Berkeley, USA.

Kristina Toutanova, Christoper D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, 3(1):83–105.

Gertjan van Noord. 2006. At Last Parsing Is Now Operational. In *Actes de la 13e Conference sur le Traitement Automatique des Langues Naturelles (TALN 2006)*, pages 20–42, Leuven, Belgium.

Gisle Ytrestoel, Dan Flickinger, and Stephan Oepen. 2009. Extracting and annotating Wikipedia sub-domains: Towards a new escience community resource. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theory*, pages 185–197, Groningen, the Netherlands.

Kun Yu, Miyao Yusuke, Xiangli Wang, Takuya Matsuzaki, and Junichi Tsujii. 2010. Semi-automatically developing chinese hpsg grammar from the penn chinese treebank for deep parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1417–1425, Beijing, China.

Yi Zhang, Stephan Oepen, and John Carroll. 2007. Efficiency in unification-based N-best parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT 2007)*, pages 48–59, Prague, Czech.