Rule Equivalence for Error Detection

Markus Dickinson

Georgetown University
Department of Linguistics
E-mail: mad87@georgetown.edu

1 Introduction and Motivation

Grammars can be induced from treebanks for a potential variety (e.g., parsing), but this task faces two major problems. One is the existence of annotation errors in the treebank which can affect the final system performance (cf. [4]). The other is that the sheer number of rules is overwhelming for most applications. As Gaizauskas [6] and Krotov et al [10] point out, in the Wall Street Journal (WSJ) corpus, as part of the Penn Treebank [12], there are over 17,000 rules for 50,000 sentences. Not only is this an efficiency issues, but the presence of more rules for rare constructions can, for example, lead to worse parsing performance [5].

To address this second problem, grammar compaction methods have been developed to reduce the size of the rule set. One general approach is to identify the essential rules and keep only those. For example, Krotov et al [10] parse the right-hand side of grammar rules and eliminate those rules which are parsable by other rules. With this method, they dramatically reduce the size of the grammar needed for parsing, but also potentially alter the structure of parses.

Another approach is to identify the essential information encoded in a rule and keep only that information. As an example of this, Hepple and van Genabith [7] compact rules by merging node categories into supertags. For instance, comparative adjective (JJR) and superlative adjective (JJS) in the WSJ are combined into a single category. While the method preserves structure, the rule growth rate is still quite severe, reducing the full 17,000 rules to only approximately 11,000 rules.

Keeping the essential information can alternatively be viewed as defining when different rules are equivalent. For this, we need some criteria by which to judge equivalence (or similarity) between rules, and these criteria might be application-dependent. The grammar compaction methods above have focused on ways to compact rules for PCFG parsing, but we are interested in annotation error detection,

Hajič, J. and Nivre, J. (eds.): Proceedings of the TLT 2006, pp. 187–198.

which addresses the first problem for grammar induction. Error detection will determine our criteria, but exploring rule similarity in this context can provide insight into grammar compaction and into techniques where rule similarity is at issue, e.g., memory-based parsing (cf. [11]), or applications which parse adverbs separately from arguments (e.g., [9]).

In order to determine what information is essential in a rule, we can ask what relevant linguistic properties are encoded by the rules. A key insight, described in section 2, is that rules can be categorized by their daughters lists; furthermore, the essential linguistic information in the daughters list is only that which is needed to predict the mother category. Thus, we compact rules in such a way as to preserve this essential information.

By exploiting simple properties of rules in order to establish equivalences between them, we can significantly decrease the amount of rules in the grammar. In section 3, we describe three incremental steps to rule compaction: 1) removal of non-essential categories (section 3.1.1), 2) mapping of head-equivalent categories (section 3.1.2), and 3) removal of duplicate material (section 3.1.3). This results in a dramatic reduction in the rule growth rate, revealing that the growth of rules which present truly new constructions is quite small. We then turn towards error detection in section 4, providing an endocentricity check on rules. All of the work is done with a small amount of manual effort, as the linguistic properties are by their nature dependent on the annotation scheme.

2 Error Detection and Daughter-Based Rules

Dickinson and Meurers [4] investigate the consistency of labeling within local trees in order to find errors, relying on the linguistic insight of endocentricity: a category projects to a phrase of the same general category. Because one can generally determine the syntactic category of the mother based on the categories of the daughters, they search for variation in the mother categories dominating the same daughters in order to find erroneous annotation in local trees. Example (1), for instance, shows how variation in labeling between adjective phrase (ADJP) and unlike coordinated phrase (UCP) for the daughters list JJ , NN CC JJ helps us detect that both instances should be UCP.

(1) a. [UCP federal/JJ ,/, state/NN and/CC local/JJ] public officials
    b. [ADJP scientific/JJ ,/, engineering/NN and/CC academic/JJ] communities

While this method is fairly accurate in finding errors, it has some shortcomings, largely in terms of recall. Many rules are rare, and a rule which occurs only once cannot be in variation with any other rule. For example, in (2), the daughters lists

ADVP RB ADVP (2a) and ADVP , RB ADVP (2b) are treated distinctly; however, the comma makes no difference with respect to what the mother is. If we treat them as the same daughters list, however, then we discover that there are two different mothers (PP and ADVP) for the same daughters. From this, we can determine that PP is not an acceptable mother in (2a) (see section 4). We now just need some way to say when two daughters lists are the same.

(2) a. to slash its work force in the U.S. , [PP [ADVP as] soon/RB [ADVP as next month]]

b. to require insiders to report their purchases and sales [ADVP [ADVP immediately] ,/, not/RB [ADVP a month later]] .

3 Determining Rule Equivalence

3.1 Using simple rule properties

To compact rules, we group daughters lists (i.e., the right-hand sides of rules) into equivalence classes in such a way that the mother is still predictable. We can use some fairly simple properties of rules for this, provided that we know how the categories in the treebank propagate properties to their mother. We use the Wall Street Journal (WSJ) portion of the Penn Treebank [12] as our test case.

3.1.1 Non-essential items

The first step in compaction identifies and removes daughter categories which are always non-essential to the categorization of the phrase. It is clear that the head of a rule is essential since head information percolates up, but we also need the head's arguments to distinguish one rule from another. Knowing that VP (verb phrase) is the head of VP NP PP, for example, does not tell us that SINV (inverted sentence) is an appropriate head; the fact that VP precedes its argument NP (noun phrase) does.

Thus, we hand-identify which items can never be heads or arguments—i.e., always function as adjuncts—and we remove them from the daughters lists. This involves three types of elimination.

Punctuation The first case is punctuation. Removing punctuation from consideration has a history in parser evaluation (see, e.g., the discussion in Hollingshead et al [8]), which highlights its unimportance in predicate-argument structure. We follow that trend by removing the following items from daughters lists: “,”, “.”, “:”, “;”, “’’”, “``”, “"”, “-LRB-”, “-RRB-”, “$”, and “#”. These items can never

usually function as modifiers and would make for removal possibilities to be removed, but it is less clear that they always function this way; it is possible for reduced relative clauses to be selected for. A category like PRT (particle) is arguably redundant, in that it always appears within verbal constructions; however, it is hard to argue on linguistic grounds that it is functioning as an adjunct, in that a verb typically selects for the particle. Thus, we keep these categories.

Empty elements The final element that we remove is the category -NONE-, simply because of its uninformativeness with respect to the mother category. -NONE- can refer to almost any category, and so its presence does not really tell us anything about the nature of the phrase.[1]

3.1.2 Head-equivalent categories

After eliminating non-essential items, we then group together lexical categories which are category equivalent. For example, singular common noun (NN) and plural common noun (NNS) are the same in how they potentially propagate nounhood. Merging them does not affect what the mother is: phrases headed by either NN or NNS are generally NP. The groupings we use are similar to the mappings used in Hepple and van Genabith [7] (specifically, their mapping #2) and are shown in figure 1. All other lexical tags (CC, CD, EX, FW, IN, LS, POS, RP, SYM, TO, UH, WP, WRG) are not grouped with any other tags.

It might be argued that some categories within a class can function differently syntactically. For example, plural nouns can appear without a determiner (DT) in a noun phrase, whereas singular nouns generally cannot. But this is not entirely true: singular mass nouns, also labeled NN, can appear without a determiner, and so we find NP → NN in the treebank. For this tagset, categories with the same "base category" tend to function in the same manner in what they select for. Crucially, there is no difference in terms of what the mother category can be.

[1] Following Gaizauskas [6], while removing -NONE-, we could merge unary structures rooted in -NONE- into a higher level of structure, for tasks such as parsing.

Base category	Mapping
Determiners	{DT, PDT, PRP$}
Adjectives	{JJ, JJR, JJS}
Nouns	{NN, NNS, PRP}
Proper nouns	{NNP, NNPS}
Adverbs	{RB, RBR, RBS}
Verbs	{MD, VB, VBD, VBG, VBN, VBP, VBZ}
Wh-determiners	{WDT, WP$}

Figure 1: Head-equivalence classes in the WSJ

We would have liked to have grouped proper nouns with other nominals, but we face the problem, pointed out in Dickinson and Meurers [4], that proper nouns do not behave in an endocentric fashion. Any word functioning as a title is labeled NNP (singular proper noun) or NNPS (plural proper noun), yet the dominating syntactic category may be VP. Thus, proper nouns were kept on their own.

3.1.3 Duplicate removal

After the first two steps have been performed, adjacent identical elements are condensed into a single element. In other words, a sequence like NN NN becomes NN. The intuition is that a series of identical categories is modeling a flat structure, which could be rendered by a rule with the Kleene + operator (XP+).

Identifying a sequence of single items is not sufficient to find all cases of flat structure labeling. In a sequence like NN IN NP IN NP, it is unlikely that the second IN NP tells us anything additional about the mother category, and it is possible that the flat structure should have more structure to it (cf. [10]). Thus, we searched for all repetitious sequences and condensed them into the shortest possible sequence. For example, JJ NN NN JJ NN becomes JJ NN because the first two NNs reduce to NN, at which point JJ NN JJ NN is reduced to JJ NN. This allows us to perform Kleene reduction on arbitrarily long sequences.

3.2 Results of Compaction on the WSJ

As an indication of the effectiveness of our equivalence classes with respect to how much generalization they accomplish, we find the following results, shown in figure 2: 15,989 daughters lists are captured by 3783 equivalence classes, i.e., 4.23 daughters lists per class. Likewise, the treebank of 17,346 rules is reduced to 4895 rules.

	Original	Compacted
Rules	17,346	4895
Daughters lists	15,989	3783
Variations	844	546

Figure 2: Results of Grammar Compaction on the WSJ

We also report the number of compacted daughters lists which have more than one mother, i.e., the number of variations. For the compacted rules, we find 546 variations. Note that although this is a smaller number than the 844 reported in Dickinson and Meurers [4] for the original rule set, we actually have found more cases. These numbers describe the variations and not the number of rules involved; because we have grouped more rules together, each one of our cases is much larger.

3.2.1 Rule growth rate

Following the evaluations in Krotov et al [10] and Hepple and van Genabith [7], we attempted to see how fast the compacted rule set grows, as compared to the original set of rules. Starting with section 00 of the WSJ, we incrementally added each section, up to section 24. We then compared the number of total rules to the number of compacted rules, as shown in figure 3.

As we can see, the growth rate for both sets of rules continues to grow indefinitely, though at much different rates. On average, for every 3.54 rules we add to the treebank, we add only one compacted rule. Not surprisingly, this shows that constructions which introduce new head and argument combinations become quite rare; at some point, they may even level off. Further analysis of the compaction process, e.g., the distance between the original daughters list and the compacted class, might reveal further ways to generalize the classes.

Compacting this way will also allow us to predict other rules which should occur (cf. [7]). Assuming an accurate set of equivalence classes, then each mother should be able to take any of the daughters lists within the class. We leave this prediction, including prevention of overgeneralization, for future work.

3.2.2 Evaluation of the Compaction Criteria

To gauge how effective our compaction criteria are, we sampled 100 compacted rules from section 00 of the WSJ corpus and examined them by hand. We found that in 98 cases, the compaction removed nothing essential for predicting the mother category. In fact, 24 of the cases were minimally compacted, needing nothing fur-

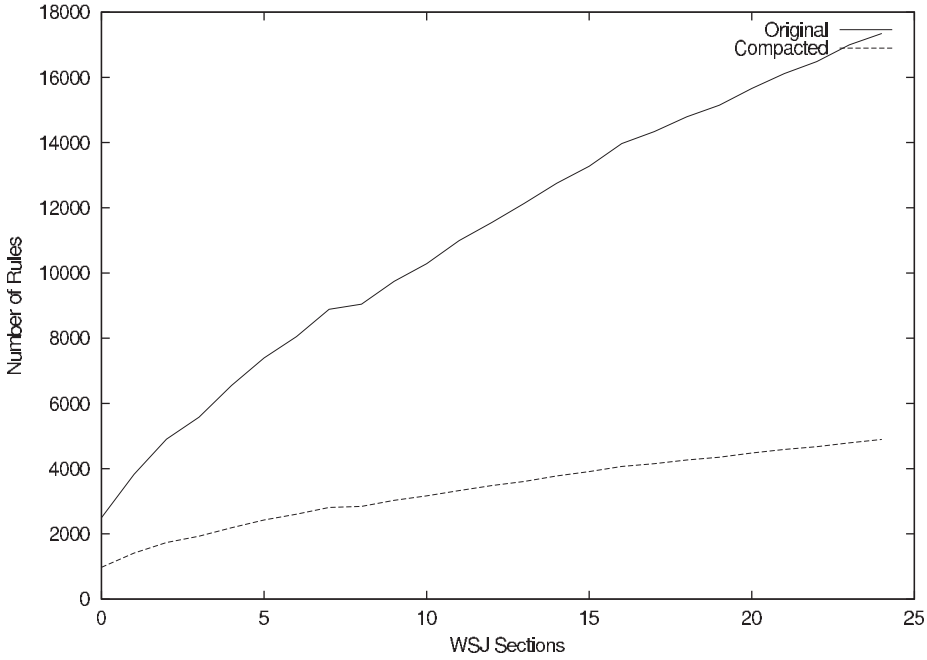

Figure 3: Rule Growth Rate in the WSJ

ther removed. The two cases which were not successful compactions involved NAC (not a constituent), as in (3). The daughters list NNP , NNP , gets compacted to NNP, but this is not a sufficient predictor of NAC as the mother: Bies et al [1] (p. 208) specifically mention that the presence of a comma determines the label NAC.

(3) [$_{NAC}$ Albuquerque/NNP ,/, N.M./NNP ,/,]

Thus, in the future we would like to revise the compaction method such that it considers full sequences of categories, as opposed to looking at categories in isolation. What we need are context-specific criteria, which say, for example, that ADJP in some instances is an argument (e.g., VBG ADJP) and in others an adjunct (e.g., DT JJR ADJP NN), and these compaction rules could be automatically determined.

Since we have hand-written the categories to remove and the head-equivalent classes, this raises the question of how portable the method is to other treebanks and languages. For a task like error detection—and likely other similarity-based tasks—we expect that such hand-written rules can be successfully employed. Of course, one has to be careful about which abstractions to make; case distinctions which are important for subcategorization should not be conflated, for example. Automating the compaction process should make the method even more general.

4 Endocentricity Check

We have grouped rules on the basis of their daughter properties, and now we can examine the properties of the mother sets, in order to find erroneous rules. Dickinson and Meurers [4] use heuristics involving frequency of rules and likelihood of being a true ambiguity to identify specific erroneous rules. These heuristics, although somewhat successful, are rather ad hoc.

4.1 Acceptable Heads of Rules

Having partially based our compactions on the notion that every rule must be headed, we use a heuristic which directly captures this property of endocentricity. To that end, we have handwritten a set of mother-daughter pairs which, different from head-finding (e.g., [3, 2]), simply indicate that a daughter is an acceptable head of a mother category. In total, we write 134 mother-head pairings—listed in appendix A—gleaned from a basic knowledge of the categories in the treebank, and we also identify eight categories which can take any daughter. Before evaluating this endocentricity check, we discuss the categories for which endocentricity is not straightforward.

Mothers with Any Daughters There were some mother categories for which we could not reasonably select a group of possible heads. We describe them here.

First, there are categories which allow elliptical constructions. Due to ellipsis, the sentence label S can be the mother of virtually anything, such as NP ADJP. Reduced relative clauses (RRC) are similar in that they generally have no verb head. Quantifier phrases (QP) can also be elliptical, as in *or more*, which is QP in a numerical context like *35 % interest or more*, despite having no numerical value within it. Secondly, categories like interjection (INTJ) and list item (LS) are simply too varied. With cases like *You see* and *C'mon* as INTJ, we cannot reliably predict the head daughter. Finally, there are categories which seem to be garbage bin categories, applicable to nearly anything: PRN, FRAG (fragment), X (uncertain, unknown, or unbracketable). In these cases, it cannot consistently be determined what the head is.

Daughters with Any Mothers On the other side, we have categories which can head any rules: -NONE- and NNP. -NONE- is a non-local placeholder and as such can take on almost any category. The proper noun tag NNP—as discussed in Dickinson and Meurers [4]—can head different categories because titles are annotated as NNP(S) in the POS annotation, but as running text (e.g., VP) in the syntactic annotation. Thus, if either is present, a potential head is present for any category.

Non-head Percolation In some cases, we do not define possible heads, but rather which daughter is mandatory for the mother category. *Wh*-phrases are arise because some *wh*-element has percolated up to this level (see chapter 9 of [1]). For example, JJ is the head of WHADJP, but it is WHADJP because of the presence of a *wh*-modifier (WRB or WHADVP); with no *wh*-modifier, it would be ADJP. Likewise, it is not clear what the heads of the categories CONJP and UCP, but they are distinguished by the fact that they must contain a conjunction (coordinating (CC) or subordinating (IN)).

4.2 Testing the heuristic

We tested this endocentricity check on the original rule set from the WSJ. For every daughters list with multiple mothers (844 variations), we want to find which mothers are erroneous.

A sample of 100 variations ([4]) points to 291 rules, 127 of which are errors (types), and they point to 847 instances of rules which are wrong (tokens). Reporting the number of erroneous types of rules found and also the total number of erroneous tokens which are detected, the results for the previous best heuristic are given in figure 4.

	Precision	Recall
Types	73.03% (65/89)	51.18% (65/127)
Tokens	65.59% (364/555)	42.98% (364/847)

Figure 4: Results of the best heuristic in Dickinson and Meurers [4]

This heuristic combines frequency information with information about the likelihood of being an ambiguity. The direct endocentricity check has a clearer interpretation, but how does it compare numerically?

	Precision	Recall
Types	78.46% (51/65)	40.16% (51/127)
Tokens	77.01% (633/822)	74.73% (633/847)

Figure 5: Evaluating the endocentricity check

The results are given in figure 5, which shows improved precision and dramatically improved token recall. The improved precision arises because we have more carefully defined what is appropriate, in a way which reflects the cause of many erroneous variations. The high token recall comes about because the most frequent incorrect rules are the most serious violations of endocentricity. For example, the

incorrect rule NP → VBG appears 177 times, reflecting a gap between the POS annotation and the syntactic annotation.

The type recall is rather low, but this is not too surprising, given that we cannot find any errors involving eight different mother categories (FRAG, INTJ, LS, PRN, RRC, QP, S, X) because we could not absolutely determine their set of possible heads. Although these cases apparently involve very few rule tokens, we could increase our type recall by combining elements of each heuristic.

Impact on Compaction Using this endocentricity check on the equivalence classes defined in section 3, we can begin to eliminate erroneous rules from consideration. Specifically, we find that removing all rules which contain endocentricity violations gives us a final rule set of 4384 compacted rules, i.e., we have eliminated 511 compacted rules.

5 Summary and Outlook

We have shown how to compact rules for error detection by grouping their daughters lists into equivalence classes, on the basis of a few simple properties which maintain the essential elements of rules. These compaction criteria were highly precise, and using them demonstrated that the rule growth of the essential elements is much less dramatic than the overall rule growth. After determining rule equivalence classes, we were also able to eliminate potentially erroneous rules by running an endocentricity check, shown to be successful on the WSJ corpus.

There are a variety of improvements that could be made to the model, some of which were pointed to in section 3.2.2. The current grouping of rules is fairly simple, and one could experiment with more sophisticated clustering algorithms for the daughters lists. Ideally, one would be able to eliminate modifiers, perhaps by identifying frequently-occurring subsequences, such as DT JJ, that occur in the same contexts as shorter subsequences without a modifier, e.g., DT. By automating as much as possible the removal of unessential items, we hope to make the process more applicable to other treebanks.

With the goal of detecting errors, we also need to perform a full evaluation of whether the equivalence classes actually contribute to error detection. In other words, of the 546 variations, how many errors do we find?

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. IIS-0623837. I would like to thank two anonymous reviewers for their helpful comments.

References

[1] Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania, 1995.

[2] David Chiang and Daniel M. Bikel. Recovering latent information in treebanks. In *Proceedings of COLING 2002*, pages 183–189, Taipei, 2002.

[3] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1999.

[4] Markus Dickinson and W. Detmar Meurers. Prune diseased branches to get healthy trees! How to find erroneous local trees in a treebank and why it matters. In *Proceedings of TLT 2005*, Barcelona, Spain, 2005.

[5] Kilian Foth and Wolfgang Menzel. Robust parsing: More with less. In *Proceedings of the Workshop on ROMAND 2006*, 2006.

[6] Robert Gaizauskas. Investigations into the grammar underlying the Penn Treebank II. Technical Report Research Memorandum CS-95-25, University of Sheffield, 1995.

[7] Mark Hepple and Josef van Genabith. Experiments in structure-preserving grammar compaction. In *1st Meeting on Speech Technology Transfer*, Seville, Spain, 2000.

[8] Kristy Hollingshead, Seeger Fisher, and Brian Roark. Comparing and combining finite-state and context-free parsers. In *Proceedings of HLT/EMNLP 2005*, Vancouver, 2005.

[9] Thomas Koller. Knowledge-based intelligent error feedback in a Spanish ICALL system. In *Proceedings of The 14th Irish Conference on Artificial Intelligence & Cognitive Science*, 2003.

[10] Alexander Krotov, Mark Hepple, Robert J. Gaizauskas, and Yorick Wilks. Compacting the Penn Treebank grammar. In *Proceedings of COLING/ACL 1998*, pages 699–703, 1998.

[11] Sandra Kübler. Towards case-based parsing: Are chunks reliable indicators for syntax trees? In *Proceedings of the COLING/ACL Workshop on Linguistic Distances*, Sydney, 2006.

[12] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

A Possible Mother-Head Daughter Pairings

Mother	Possible Heads
SBAR	NNP, -NONE-, VP, S, SBAR, SBARQ, SINV, SQ
SBARQ	NNP, -NONE-, SBARQ, SQ
SINV	NNP, -NONE-, MD, VB, VBD, VBG, VBN, VBP, VBZ, VP, SINV
SQ	NNP, -NONE-, VB, VBD, VBG, VBN, VBP, VBZ, VP, S, SQ

Figure 6: Clause level head restrictions

Mother	Possible Heads
ADJP	NNP, -NONE-, CD, JJ, JJR, JJS, ADJP, QP
ADVP	NNP, -NONE-, RB, RBR, RBS, ADVP
CONJP	NNP, CC, IN
NAC	-NONE-, CD, DT, JJ, JJR, JJS, NN, NNP, NNPS, NNS, PRP, NP, QP
NP	-NONE-, CD, EX, DT, JJ, JJR, JJS, NN, NNP, NNPS, NNS, PRP, NP, QP
NX	-NONE-, CD, DT, JJ, JJR, JJS, NN, NNP, NNPS, NNS, PRP, NX, QP
PP	NNP, -NONE-, IN, TO, PP
PRT	NNP, RP
UCP	NNP, CC, IN
VP	NNP, -NONE-, MD, VB, VBD, VBG, VBN, VBP, VBZ, VP
WHADJP	NNP, -NONE-, WRB, WHADVP, WHADJP
WHADVP	NNP, -NONE-, WRB, WHADVP
WHNP	NNP, -NONE-, WDT, WP, WP$, WRB, WHADJP, WHADVP, WHPP, WHNP
WHPP	NNP, -NONE-, WHNP, WHPP

Figure 7: Phrase level head restrictions