Similarity Rules!
Exploring Methods for Ad-Hoc Rule Detection

Markus Dickinson
Indiana University
Department of Linguistics
md7@indiana.edu

Jennifer Foster
Dublin City University
School of Computing
jfoster@computing.dcu.ie

1 Introduction and Motivation

One problem facing the extraction of treebank grammars is that of *ad hoc* rules, rules used for constructions specific to one data set and unlikely to be used on new data (Dickinson, 2008). These rules can be erroneous, cover ungrammatical text, or reveal issues with the treebank's annotation scheme. These are significant problems since training on erroneous data can be detrimental to parsing performance (e.g., Dickinson and Meurers, 2005; Hogan, 2007), and the use of precision grammars in grammar checking and generation requires distinctions between grammatical and ungrammatical sentences (e.g., Bender et al., 2004). Ad hoc rules are especially problematic when they point to inconsistent aspects of the annotation scheme, as the scheme forms the basis of any analysis using it.

Together with these problematic cases, there is also the practical issue of determining which treebank rules are useful for parsing new data: ad hoc rules can also simply be rules that do not generalize. Although frequency is often used to determine generalizability, this is not unproblematic, as low-frequency rules can be valid and potentially useful rules (e.g., Daelemans et al., 1999); high-frequency rules can be erroneous (e.g., Dickinson and Meurers, 2005); and frequency is not completely related to usefulness for parsing (e.g., Foth and Menzel, 2006). Furthermore, infrequent rules in one genre may be quite frequent in another (Sekine, 1997). Thus, methods are needed to determine which rules are generalizable to new data or not.

The previous detection of ad hoc rules, while effective, could be improved in several ways. First, it currently relies on treebank-specific knowledge,

such as knowing which part-of-speech categories were *equivalent* (Dickinson, 2008). If we can justify removing the dependence on corpus-specific properties, the methods can be made more corpus-independent.

Additionally, it is not clear whether the key properties identifying ad hoc rules are based on frequency, some other criterion—namely similarity to other rules—or both. Understanding the contribution of similarity is crucial if we are to move beyond using ad hoc rule detection simply for treebanking and into realms which already use frequency information, such as parsing.

Finally, the methods have only been developed to analyze rules which occurred in some observed data. Although this is useful for examining rule generalizability, it does not tell us anything about the quality of the rules that never occurred in that data. Identifying the quality of unseen rules can begin to point to how one might generalize a grammar to cover new types of data, for cross-genre parsing (cf. Gildea, 2001; McClosky et al., 2006).

We thus examine the role of similarity in ad hoc rule detection and show how previous methods can be made more corpus independent and more generally applicable. After reviewing the previous methods in section 2, we turn to the rationale for similarity-based rule comparison without any treebank-specific knowledge in section 3.1. As we will see, the idea that similarity to other rules indicates the reliability of a rule does not require any corpus-specific information. Building on this, we discuss the component parts of rule comparison, namely frequency and similarity, in section 3.2 and outline how to adapt the methods for unseen rules in section 3.3. We evaluate the methods in section 4, showing that the revised metrics for calculating ad hoc scores are as good as, if not better than, the previous ones, while being more general, providing a foundation for grammar generalization.

2 Background

Dickinson (2008) detects ad hoc rules by first grouping rules into equivalence classes and then looking for similar rules across the grammar. Rules with the same mother are grouped into equivalence classes by the following steps:

1. Remove daughter categories that are always non-predictive to phrase categorization, i.e., always adjuncts, such as punctuation.
2. Group head-equivalent lexical categories, e.g., NN (common noun) and NNS (plural noun).

Then, two methods are used to detect ad hoc rules. The first method (*whole daughters scoring*) directly accounts for similar rules across equiva-

lence classes. Each rule type is assigned a *reliability score* by adding 1 for every rule token within the equivalence class and adding $\frac{1}{2}$ for every rule token in a highly similar equivalence class. Rules with the lowest scores are flagged as potentially ad hoc.

To determine similarity, a modified Levenshtein distance is used, where only insertions and deletions are allowed; a distance of one qualifies as highly similar. Substitutions are not used, as they are too problematic to include, Consider the erroneous rule VP → RB, which occurs once in the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993). With substitutions, there would be 760 "comparable" instances of VP → VB, despite the vast difference in category (verb [VB] vs. adverb [RB]).

The other method of detecting ad hoc rules (*bigram scoring*) calculates reliability scores by abstracting a rule to its bigrams and examining which bigrams the classes do not have in common. Using equivalence classes in the calculations, both methods effectively identify ad hoc rules. Since the whole daughters method is more effective across a range of tests, we focus on that method in this paper, although our work is applicable to both.

3 Corpus-independent similarity

3.1 Corpus-independence

The notion of similarity above, namely the comparison to whole lists of daughters, is a useful way to compare rules, but this is done only after rules have been reduced into equivalence classes. While the use of equivalence classes is well-motivated—e.g., it generally makes no difference whether a rule has punctuation or not—this is not always ideal.

First, the groupings into head-equivalent categories are not always sound. For example, JJ (adjective) and JJR (comparative adjectives) generally predict the same mother (ADJP) and generally modify nouns. However, there are syntactic differences. As an example, consider (1): comparative adjectives are used in correlative *the*-clauses (Bies et al., 1995, p. 303ff), whereas positive adjectives are generally not. This is a syntactic context where the two categories are not replacable; treating them separately captures this.

(1) The sooner our vans hit the road ... [$_X$ the/DT easier/JJR] it is for ...

Secondly, by comparing rules to similar rules, we are already naturally capturing equivalences among rules. The categories JJ and JJR often are replacable, which we can tell by the observable fact that they behave similarly

with respect to other comparable rules. For instance, NP → DT JJ NN (9866 tokens) and NP → DT JJR NN (234 tokens) are comparable rules because they both are one step away from NP → DT NN (29,217 tokens). Grouping rules around the more basic rule NP → DT NN already indicates similar properties, without requiring pre-specification.

If we can remove the corpus-dependent rules for forming equivalence classes and achieve comparable results, we prefer to do so, in order to be able to work on a variety of corpora without learning the annotation scheme. In this paper, we thus remove the criteria for forming equivalence classes, and calculate reliability scores, using rules as they are. These equivalences, though, still provide a useful goal: Levenshtein distance is effective as a means for determining similarity because it captures properties such as the removability of adjuncts and the natural equivalence of categories.

By more automatically creating equivalences between rules, we are more sensitive to the surrounding categories. For example, a category may always be an adjunct, but it can still be important to know that the adjunct exists within a certain rule. Consider (2a), with the rule ADVP → RB RB -LRB- PP -RRB- PP. Although both -LCB- and -RCB- (codes for left and right curly brackets) are adjuncts, and labeled as such (-LRB-, -RRB-), observing them within another structure is odd, given that the preferred analysis is to embed such bracketed material, as in (2b). We see here that even adjuncts, such as brackets, provide useful information about a rule being ad hoc.

(2) a. . . . they try * to build it [$_{ADVP}$ somewhere/RB else/RB -LCB-/-LRB- [$_{PP}$ in Europe] -RCB-/-RRB- [$_{PP}$ besides the U.K.]] ,

b. [$_{ADVP}$ somewhere/RB else/RB [$_{PRN}$ -LRB- in Europe -RRB-] [$_{PP}$ besides the U.K.]]

3.2 Contributions of information

The methods are based on two major components, frequency of a rule and its similarity to other rules. Removing the equivalence class restriction, as outlined above, we still have the following as our method for computing the whole daughters score, which we use for *reliability scores* in this paper:

1. For every identical rule token, add 1.
2. For every highly similar rule token, add $\frac{1}{2}$.

It is clear from this formulation that we can split the calculation into a frequency score (#1) and a similarity score (#2). Given that frequency

is already something which has been used effectively in parsing models, we need to more fully investigate the role that similarity plays in the detection of ad hoc rules, as it has the potential to provide more fine-grained information not available from frequency. When we calculate *similarity scores* on their own, we simply count one for each similar rule (instead of using $\frac{1}{2}$), as this has a more natural interpretation as the number of similar rules.

3.3 Unseen rules

The method currently only calculates scores for rules which occurred in one data set, namely the training data. But any rule in a set of heldout data can be compared to see how similar it is to the rules in the grammar from the training data. The scoring outlined above can be used, with the only difference that the identical and similar rules are in a different data set. For example, in (3), we have two rules which did not appear in our training data (see section 4); in the case of NP → DT RB PRN S (3a), we have a similarity score of 0, as no rules in the training data are similar, and the rule is incorrect. In the case of the unseen and correct NP → JJ " JJ NN (3b), we have a similarity score of 866, i.e., 866 similar rules.

(3) a. A put option gives its holder [$_{NP}$ the/DT right/RB [$_{PRN}$ but not the obligation] [$_{S}$ * to sell a stock ...]]

b. see [$_{NP}$ various/JJ "/" unmet/JJ needs/NN] , "/" ...

When examining a data set, then, there are two ways one can detect ad hoc rules. On the one hand, one can calculate similarity scores compared to other rules in the same data set. This indicates which rules are inconsistent with other rules in the same grammar and is thus useful for treebanking. On the other hand, one can calculate similarity scores compared to a different data set, indicating which rules are more likely to have emerged from a different grammar. These rules not only might be ad hoc, but they might be indicative of a unique type of construction, whether because the genre is different or because someone else annotated the data.

4 Evaluation

4.1 Unreliability scores

To evaluate, we follow Dickinson (2008) and see how well the scores combining frequency and similarity—the reliability scores—predict a rule's "ungeneralizability." We train on sections 2-21 of the WSJ, and evaluate on

section 24. Table 1 shows the ungeneralizability rate for the whole daughters method, for different thresholds.

Threshold	Rules	Unused	Ungeneralizability
1.0	1625	1617	99.51%
2.0	2801	2785	99.43%
3.0	3515	3479	98.97%
4.0	4011	3965	98.85%
5.0	4412	4357	98.75%

Table 1: New whole dtr. ungeneralizability (WSJ-24)

The results are quite good and dramatically surpass the values presented in Dickinson (2008). For example, a threshold of 50 in the previous whole daughters method with equivalence classes identifies 3548 rules with 96.93% ungeneralizability. For that same approximate number of rules (threshold=3.0), the method without equivalences has 98.97% precision.

Testing across genre Table 2 shows the results of evaluating the whole daughters method on a new set of 1,000 gold standard parse trees (Foster and van Genabith, 2008). The sentences were taken from the British National Corpus (Burnard, 2000), and each was selected on the basis that it contains a verb which does not appear in the WSJ training data. Because there are only 1,000 hand-corrected parse trees, we perform a five-fold cross-validation with training/test splits of 800/200. Again, abandoning the use of equivalence classes appears to be effective. At a threshold of 50.0, the whole daughters method making use of equivalence classes has a precision of 88.51% for 790 rules, whereas, for a similar number of rules (threshold of 5.0), the new method has a precision of 92.52%. At a threshold of 35.0, the old method has a precision of 88.59% for 708 rules, whereas for a threshold of 3.0, the new method has a significantly higher precision of 94.14%.

Threshold	Rules	Unused	Ungeneralizability
1.0	388	376	96.91%
2.0	585	559	95.56%
3.0	683	643	94.14%
4.0	758	707	93.27%
5.0	802	742	92.52%

Table 2: New whole dtr. ungeneralizability (BNC1000, five-fold)

To better see what happens with different genres, we also examined what happens when we train on the original training data (WSJ2-21)[1] and evaluate on the BNC1000. For approximately 1,600 rules (new whole daughters threshold of 1.0, old of 8.0), the new method without equivalence classes does a better job (99.25% vs. 98.92%), and for approximately 4,300 rules (new threshold of 5.0, old of 81.0), the new method achieves a precision of 98.66% versus 96.84% for the old method. These results all indicate that the whole daughters method without equivalence classes is superior to the method with them, in terms of predicting which rules are useful for new text. The results are even comparable across genres.

4.2 Similarity vs. Frequency

Based on the previous results, the methods identify ungeneralizable rules quite accurately. The remaining question is: to what extent is this an effect of frequency and to what extent is this an effect of similarity?

Threshold	Rules	Unused	Ungeneralizability
1	8776	8627	98.30%
2	10,741	10,475	97.52%
3	11,601	11,253	97.00%
4	12,131	11,723	96.64%

Table 3: Frequency ungeneralizability (WSJ-24)

Frequency ungeneralizability As we can see in table 3, frequency on its own is a solid indicator of a rule's ungeneralizability, accurately identifying thousands of rules which do not appear in the development data. However, in identifying so many rules, it is rather coarse, not allowing us to sort infrequent but useful rules from infrequent but problematic rules.

Similarity ungeneralizability For the similarity scores (i.e., number of similar rules without including the frequency of the rule in question), we can see in table 4 that whole daughters scoring is also effective at identifying ungeneralizable rules. More than that, this scoring is providing information more fine-grained than that available from frequency. Most of the 32 rules with 0 scores which appear in the new data are fairly frequent (9 occur more

[1]Since the BNC1000 does not contain traces, these first had to be removed.

Threshold	Rules	Unused	Ungeneralizability
0	1851	1819	98.27%
1	2622	2571	98.05%
2	3147	3080	97.87%
4	3865	3769	97.52%

Table 4: Whole dtrs. similarity (WSJ-24)

than 100 times in the training data and are thus clearly reliable), which is why the reliability scores, combining frequency and similarity, work best.

Consider also that 1625 of the 1851 identified rules are single-occurrence rules. Of these, 1617 (99.51%) do not generalize to the development data. In fact, for all rules with frequencies of 1, if the similarity score is 2 or less, the ungeneralizability rate is 99.48% (2661/2675), and for similarity scores of 10 or less, it is 99.17% (4085/4119), higher than the 98.30% for all single-occurrence rules (table 3). It thus seems that, for low-frequency rules, similarity scores can sort rules within their frequency class.

4.2.1 Unseen rules

Taking into account rules in the evaluation data which did not occur at all in the training data, we can more fully evaluate how well similarity scores extend to new data. Table 5 shows how the scores change, once unseen rules are also included in the evaluation, dropping dramatically (cf. table 4).

Threshold	Rules	Unused	Ungeneralizability
0	1952	1819	93.19%
1	2747	2571	93.59%
2	3290	3080	93.62%
4	4033	3769	93.45%

Table 5: Whole dtrs. similarity, with unseen rules (WSJ-24)

The biggest factor is that 133 rules that occur in the heldout data have a score of 0, and 101 of these never appeared in the training data. Interestingly, however, this is out of 396 rules in WSJ-24 which have a frequency of 0 in the training data. In other words, although 0% of the unseen rules are predicted by frequency alone, 295, or 74.5%, of the new rules have a similarity score greater than zero. This indicates that similarity scores have potential in providing information for grammar generalization to new data. In this case,

for example, a similarity score above 0 misses 133 rules that we need, but it picks up an additional 295. Future work can work on generalizing the grammar in such a way as not to overgeneralize.

Likewise, there are 634 rules in the BNC1000 which do not appear in the WSJ2-21 training data. A similarity score of 0 identifies 259 rules, of which 225 were not in the WSJ data. This means that 409, or 64.5%, of the unseen rules have a score greater than 0. Again, we miss 259 rules, but gain 409 by looking at similarity scores above a threshold of 0.

4.3 Analyzing rare rules

To determine the effectiveness of the similarity scores on isolating structures which are not linguistically sound, as opposed to simply identifying ungeneralizable rules, we sample 100 WSJ rules occurring only once in the training data. Without examining the rule scores, we mark each as an error, ungrammatical, unclear, or correct. Of these 100, 21 are errors, and 5 covered ungrammatical constructions. For the bottom quarter of cases identified by the original methods, the whole daughters (similarity scores ≤ 24) method finds 7 errors, or 33% of them, additionally finding 2 ungrammatical cases. For the new method which does not use equivalence classes, we find essentially the same results: the bottom 22 cases (scores $= 0$) contain 8 errors, as well as 3 ungrammatical structures.

100 BNC rules which occur once in the WSJ training material were also sampled: 30 of these contain an annotation error, 46 are annotated correctly and the remaining 24 are unclear cases, often idiomatic phrases, inadequately covered by the Penn Treebank guidelines. As with the WSJ data, the similarity scores are reasonably well aligned with the erroneous cases: the bottom third of cases with the original method (similarity scores < 3) contain 13 of the 30 erroneously annotated rules. The bottom third of cases (similarity score=0) with the new method contain 14 of these 30 rules.

We can thus see that similarity-based scores are, in practice, effective at sorting problematic low-frequency rules from less problematic ones. Taken together with the results from the previous section that similarity sorts low-frequency rules in terms of ungeneralizability, we can see that similarity is a useful feature for determining a treebank grammar.

Where similarity does not work But what about the errors that have high similarity scores? It turns out that these types of rules are those which happened to be errors, but which actually license legitimate structures. For example, the structure in (4) seems to be erroneous because *just* modifies the

NP. However, S → ADVP NP PP (score of 154) is a potentially legitimate structure: S → NP PP is licensed in the case of sentence gapping and small clauses (Bies et al, p. 127, 252ff), and a sentential adverb (ADVP) would attach as a sister of the NP and PP (Bies et al, p. 13).

(4) And the company is certain * to get out some aircraft with [$_S$ [$_{ADVP}$ just] [$_{NP}$ supervisors and other non-striking employees] [$_{PP}$ on hand]] .

Equivalences classes With this test-bed of cases, we can also analyze how the method changes by removing the equivalence class criteria, in both positive and negative ways. Starting with rules which are errors, we can see how they are more appropriately receiving lower scores. Consider example (5), with the rule NP → JJS JJ NN VBZ. The POS category VBZ is clearly wrong, but the original whole daughters method assigns it the similarity score 1,547, because the reduced rule NP → JJ NN VB is similar to lots of NP → JJ NN rules. By keeping the rule distinct, the new whole daughters score is 7, because this exact sequence is difficult to match.

(5) [$_{NP}$ most/JJS Western/JJ air/NN fleets/VBZ]

Consider also (6), with the rule NP → NP -LRB- NP , NP -RRB-. The whole daughters score goes from 10,462 to 0. The equivalence mappings would reduce this rule to NP → NP NP NP, thus losing crucial information about how bracketing should be done for parenthetical information. Although parentheses are "always adjuncts," they are informative adjuncts.

(6) [$_{NP}$ [$_{NP}$ Rep. Ronnie Flippo] -LRB-/-LRB- [$_{NP}$ D.] ,/, [$_{NP}$ Ala.] -RRB-/-RRB-] , one of the members of the delegation , says ...

From the BNC data, consider cases like (7a), which contains the erroneous rule NP -> " NN " (with the corrected version in (7b)). With the old method, this is reduced to the extremely frequent NP -> NN, resulting in a similarity score of 10,802. Without equivalence classes, it receives a score of 8. The new method proves useful in identifying the incorrect annotation of quotations.

(7) a. [$_{NP}$ [$_{NP}$ "/" wardrobe-woman/NN "/"] [$_{PP}$ at the school]]
b. [$_{NP}$ "/" [$_{NP}$ wardrobe-woman/NN] "/" [$_{PP}$ at the school]]

There are cases, on the other hand, in which the new method performs less well, assigning low scores to correct rules. For instance, in example (8) from WSJ-24, with the correct rule S → -LRB- " NP " VP . -RRB-, the

similarity score moves from 159,444 to 0. We see such a dramatic difference because of punctuation. The reduced rule was S → NP VP, which is clearly correct and similar to other rules.

(8) [$_S$ -LRB-/-LRB- "/" [$_{NP}$ Quest for Fire] "/" [$_{VP}$ was the first time] ./. -RRB-/-RRB-]

While removing some punctuation may assist in comparing rules, it is not clear-cut, as the punctuation that is meaningful can differ across treebanks. And as cases like (6) and (7a) show, punctuation can be informative.

5 Summary and Outlook

We have furthered the work of ad hoc rule detection by making it more corpus-independent. We have also verified that similarity is a crucial factor in determining the reliability of a rule, providing information unavailable in frequency, including a way to score rules which are not in the training data.

Given the success of such a method, a next step is to verify its effectiveness on other treebanks. An additional question is to see what effect these scores have on parser training, either by filtering rules identified by such methods, or using them in a parse reranking model . Since this work points to ways to generalize a grammar to new data, one can also explore the effects of the scores on parsing across genres (Sekine, 1997; Gildea, 2001; McClosky et al., 2006) and their applicability to active learning techniques (e.g., Tang et al., 2002).

Acknowledgements

Thank you to the three anonymous reviews for their helpful comments.

References

Bender, Emily M., Dan Flickinger, Stephan Oepen and Timothy Baldwin (2004). Arboretum: Using a Precision Grammar for Grammar Checking in CALL. In *Proceedings of the InSIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems*. Venice, Italy.

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

Burnard, Lou (2000). *User Reference Guide for the British National Corpus*. Tech. rep., Oxford University Computing Services.

Daelemans, Walter, Antal van den Bosch and Jakub Zavrel (1999). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning* 34, 11–41.

Dickinson, Markus (2008). Ad Hoc Treebank Structures. In *Proceedings of ACL-08*. Columbus, OH.

Dickinson, Markus and W. Detmar Meurers (2005). 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.

Foster, Jennifer and Josef van Genabith (2008). Parser Evaluation and the BNC: Evaluating 4 constituency parsers with 3 metrics. In *Proceedings of LREC 2008*. Marrakech, Morocco.

Foth, Kilian and Wolfgang Menzel (2006). Robust Parsing: More with Less. In *Proceedings of ROMAND 2006*.

Gildea, Daniel (2001). Corpus Variation and Parser Performance. In *Proceedings of EMNLP-01*. Pittsburgh, PA.

Hogan, Deirdre (2007). Coordinate Noun Phrase Disambiguation in a Generative Parsing Model. In *Proceedings of ACL-07*. Prague, pp. 680–687.

Marcus, M., Beatrice Santorini and M. A. Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 313–330.

McClosky, David, Eugene Charniak and Mark Johnson (2006). Reranking and Self-Training for Parser Adaptation. In *Proceedings of COLING-ACL-06*. Sydney, pp. 337–344.

Sekine, Satoshi (1997). The Domain Dependence of Parsing. In *Proceedings of ANLP-96*. Washington, DC.

Tang, Min, Xiaoqiang Luo and Salim Roukos (2002). Active Learning for Statistical Natural Language Parsing. In *Proceedings of ACL-02*. Philadelphia, pp. 120–127.