Linguistic **I**ssues in **L**anguage **T**echnology – **LiLT**
Submitted, April 2011

Detecting Ad Hoc Rules for Treebank Development

Markus Dickinson

Published by CSLI Publications

LiLT volume 4, issue 3 April 2011

Detecting Ad Hoc Rules for Treebank Development

Markus Dickinson, *Indiana University*

Abstract

We outline a method of detecting ad hoc, or anomalous, rules in treebank grammars, by exploiting the fact that such rules do not fit with the rest of the grammar. Ad hoc rules are rules used for specific constructions in one data set and unlikely to be used again. These include ungeneralizable rules, erroneous rules, rules for ungrammatical text, and rules which are not consistent with the rest of the annotation scheme. Based on the idea that valid rules should receive support from other rules in the grammar, we develop two methods for detecting ad hoc rules in flat treebanks and show they are successful in detecting such rules. Although one can put some linguistic knowledge into determining rule similarity and dissimilarity, the methods work best by using a simple, modified Levenshtein distance. We illustrate this on the English Wall Street Journal treebank and the German TIGER treebank. For the latter, we extend the method to formalisms incorporating discontinuous constituents, employing CFG-like rules for the comparisons.

LiLT Volume 4, Issue 3, April 2011.
Detecting Ad Hoc Rules for Treebank Development.

1 Motivation

When extracting rules from treebanks, especially constituency-based treebanks employing flat structures, grammars often limit the set of rules (e.g., Charniak, 1996), due to the large number of rules (Krotov et al., 1998) and "leaky" rules that can lead to mis-analysis (Foth and Menzel, 2006). Although frequency-based criteria are often used, these are not without problems because low-frequency rules can be valid and potentially useful rules (Daelemans et al., 1999), and high-frequency rules can be erroneous (Dickinson and Meurers, 2005b). A key issue in determining the rule set is rule generalizability: are these rules needed to analyze (new) data? Another way of phrasing this is: are these rules worthy of being included in a treebank grammar? We address the question of how to use information contained within a basic treebank grammar to better determine the linguistic quality of each rule.

Note the range of applicability of examining grammar rule quality. On the one hand, the rules in question may be a part of the internal representation of a parser—that is, the rule set. Thus, determining good/bad rules can be a part of rule filtering (Charniak, 1996)—thereby improving parser efficiency—or parse reranking (e.g., Hall and Novák, 2005, Charniak and Johnson, 2005). On the other hand, the rules in question may simply be part of an annotated corpus—whether hand-annotated or automatically-parsed (van Noord and Bouma, 2009). As pointed out in Dickinson (2010), identifying errors in automatically-parsed data, for annotators to then post-edit, is one way to combat the bottleneck of obtaining larger treebanks in a wider range of text types. In either case, we are asking whether the rules from *some grammar* fit the rules we observe in the data. If we think about these scenarios, we have: a) a grammar extracted from an annotated treebank, being compared to: b) individual rules of interest (in a corpus or a parser), in order to determine whether the rules fit within that grammar.

This issue of grammar rule inclusion is of even more importance when considering issues of grammar robustness, such as the task of porting a parser trained on one genre to another genre (e.g., Gildea, 2001, McClosky et al., 2006, Rimell and Clark, 2008). Infrequent rules in one genre may be quite frequent in another (Sekine, 1997) and their frequency may be unrelated to their usefulness for parsing (Foth and Menzel, 2006). Indeed, some parsing experiments alter the treebank grammar for parsing text in a new domain (e.g., Foster, 2010). Taking all of these points together, we thus define methods for detecting so-called *ad hoc* rules, rules used for particular constructions specific to

one data set and unlikely to be used on new data. We focus in this paper on *hand-annotated corpora*, but believe the methods and perspective to be applicable to the other tasks mentioned above.

Rules which do not extend to new text do so for a variety of reasons, not all of which can be captured strictly by frequency. While there are simply phenomena which are rarely used because they represent language which is rarely used (e.g., long coordinated lists), other "ungeneralizable" phenomena are potentially more troubling. For example, when ungrammatical or non-standard text is used, treebanks employ rules to cover it, but do not usually indicate ungrammaticality in the annotation. These rules are only to be used in certain situations, e.g., for typographical conventions such as footnotes, and pose a problem if the set of treebank rules is intended to accurately capture the grammar of a language. This is true in the case of precision grammars for grammar checking and generation (e.g., Wagner et al., 2007, Bender et al., 2004), and in applications like intelligent computer-aided language learning, where learner input is parsed to detect what is correct or not (e.g., Metcalf and Boyd, 2006, Dickinson and Lee, 2009). If one simply wants to capture a better theoretical model of language, it is also important to identify questionable rules, but even for robust parsing, it seems advantageous to know that some rules work only in certain situations.

Detecting ad hoc rules can also reveal issues related to rule quality, in terms of errors and annotation scheme definitions. Many ad hoc rules are simply erroneous. Not only are errors inherently undesirable for obtaining an accurate grammar, but training on data with erroneous rules can be detrimental to parsing performance (e.g., Dickinson and Meurers, 2005b, Hogan, 2007), and parser evaluation becomes more difficult to assess (e.g., Padro and Marquez, 1998, Habash et al., 2007). Other ad hoc rules point to non-uniform or inconsistent aspects of the annotation scheme. Thus, identifying ad hoc rules can also provide feedback on annotation schemes, an especially important step if one is to use the treebank for specific applications (see, e.g., Vadas and Curran, 2007), or if one is in the process of developing a treebank.

Although statistical techniques have been employed to detect anomalous annotation (Ule and Simov, 2004, Eskin, 2000), these methods do not account for linguistically-motivated generalizations across rules, and no thorough evaluation has been done on a treebank. Our starting point for detecting ad hoc rules is also that they are dissimilar to the rest of the grammar, and we use a notion of valency (cf., dependencies) to drive the grammar rule comparisons. In section 2, we outline the essential properties of valency and how they are encoded in flat

structures. From this, we outline some basic notions of dissimilarity in section 3, relying on the fact that valid rules should receive support from other rules in the grammar, and we present quantitative and qualitative evaluation in section 4. Because the notions of dissimilarity rely upon a general notion of valency, they can be applied to treebanks which do not conform to the traditional context-free grammar (CFG) formalism, namely treebanks with discontinuous constituents (section 5). We evaluate the methods for discontinuous formalisms in section 6, before turning to related work in section 7. A major contribution of this work, in addition to outlining techniques to detect ad hoc rules, is the qualitative analysis, which can provide insight into future treebank development. Additionally, looking at rule similarity is a task that is applicable to domains beyond parsing, such as product attribute candidate extraction in sentiment analysis (Zhao et al., 2010).

Parts of this work have appeared in Dickinson (2006, 2008, 2009) and Dickinson and Foster (2009), but have never been discussed as a unified whole. We here highlight the theoretical issues involved across different scenarios and in many parts provide more extensive evaluation.

2 A starting point: valency in flat treebanks

Our starting point for making rule comparisons is that of *valency*, by which we mean the complete list of a head and its arguments and adjuncts (Przepiórkowski, 2006): do rules have similar valencies or not? Valency—in this sense, a list of dependents—is a core notion of what defines a grammar. Certain valencies are well-formed and others are not, or at least are less likely, leading us to expect consistency across the set of valencies. While it is clear that dependency representations capture valency as a matter of course (Dickinson, 2010), in this paper we examine constituencies. We must thus show that: a) valencies are obtainable from constituency treebanks and b) the grammars found in treebanks are in need of improvement with respect to valency.

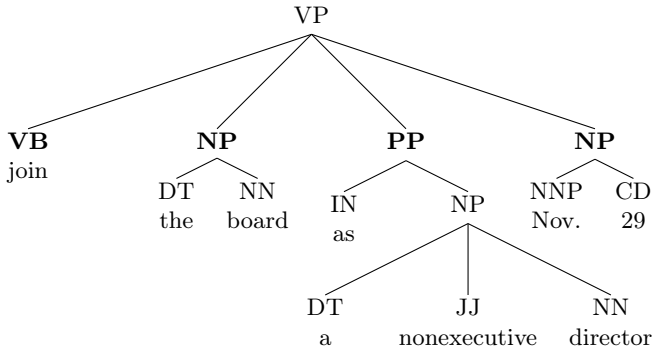
Every treebank encodes some notion of syntactic theory, even if it is only the generally agreed-upon properties of a language (see, e.g., Rambow, 2010), and there are competing factors in what kind of (descriptive) linguistic theory is encoded (cf. Elworthy, 1995, Déjean, 2000). On the one hand, corpus annotation is guided by external criteria: do the distinctions capture linguistic properties needed for certain corpus uses, such as parsing or linguistic searching? On the other hand, there are internal criteria: can the distinctions can be annotated easily and automatically?

Treebanks used for statistical parsing often emphasize broad cov-

erage and thus internal criteria, making sure that the annotation can be done consistently, with high inter-annotator agreement (e.g., Voutilainen and Järvinen, 1995). This can be at the expense of true grammar development, however. While the treebank is annotated quickly and in a way which lends itself to parsing, it is not always clear what the properties are of the encoded grammar.[1]

As one example, consider the partial tree in (1) from the Wall Street Journal (WSJ) corpus portion of the Penn Treebank (PTB, Marcus et al., 1993), which we will use throughout the paper. To avoid disagreements between annotators, the trees are given relatively flat structures; in this case, no one has to decide where the *as* prepositional phrase (PP) attaches, instead including it simply as a daughter of the verb phrase (VP). Thus, we wind up with rules like VP → VB NP PP NP, which do not correspond to theories distinguishing arguments from adjuncts in the syntactic structure. While a potential drawback, an advantage of this rule is that all the dependents of the head VB (baseform verb) are found as sisters to the head, i.e., all in one place, discussed more below.

(1)



Although such treebanks have served to advance the state-of-the-art in computational linguistics, there are reasons for investigating the grammars extracted from them. First, as illustrated above, treebanks commonly contain rather flat structures and coarse categories. This means that there are missing linguistic decisions, which have to be recovered for linguistic searching or even parser training (cf. Klein and Manning, 2003, Petrov et al., 2006). Secondly, there is the sheer number of rules to contend with. The WSJ, for example, has over 17,000 rules for 50,000 sentences. Grammar compaction methods can reduce the

[1]Notable exceptions are treebanks built in tandem with a grammar (e.g., Oepen et al., 2004, Rosén et al., 2005, Bond et al., 2004).

size of the rule set (Krotov et al., 1998, Hepple and van Genabith, 2000), but, as mentioned in the introduction, there is still a need to sort useful rare constructions from unhelpful ones (Foth and Menzel, 2006, Daelemans et al., 1999). Finally, there is the problem of annotation errors, which arise in the process of creating a large corpus. These errors can have a detrimental effect on the training and evaluation of natural language processing (NLP) systems (cf. Dickinson and Meurers, 2005a, Hogan, 2007) and also on the precision and recall for finding desired linguistic constructions (cf. Meurers, 2005). For example, Padro and Marquez (1998) show that, for many current comparative evaluation situations, one cannot truly tell which technology among competing ones (e.g., POS taggers) is better.

Practically speaking, then, the grammars underlying flat treebanks are worthwhile to inspect, as we know they are in need of refinement. But the flat representation has a benefit, in that it encodes a great deal about valency. In the above example (1), for instance, we know that the VP contains 4 items which serve to form a whole syntactic and semantic unit. For other representations, such as binary-branching treebanks, such information can be obtained through traversing a tree—provided that the head is identified—and thus this work is applicable to treebanks with a richer amount of information. However, grammars for flat treebanks can serve as an excellent starting point for defining the consistency of a grammar.[2]

3 Defining dissimilarity

3.1 Background

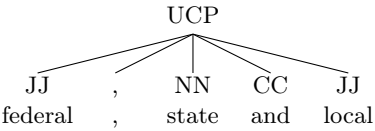
Our starting point for comparing rules comes from a method of annotation error detection which searches for inconsistency of labeling within local trees (Dickinson and Meurers, 2005b). This method is useful, in that it revolves around the question of what properties of a rule are expected to be consistent. As we will demonstrate, this leads to finding equivalency between rules.

The method is based on the fact that one can generally determine the syntactic category of the mother of a rule based on the categories of its daughters. In other words, linguistic phrase structure rules tend to be endocentric (cf. X-bar syntax, Jackendoff, 1977). Thus, Dickinson and Meurers (2005b) search for variation in mother categories which dominate the same daughters; daughters lists with more than one mother are flagged as potential errors.

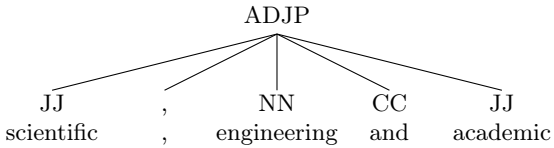
[2]For a discussion of portability of the method to richer treebanks, see section 5.2 of Dickinson (2009).

As an example, consider the daughters list JJ , NN CC JJ, which varies in the WSJ between unlike coordinated phrase (UCP) and adjective phrase (ADJP), as in (2). Here, there is indeed an error, as there is no need for variation: the guidelines indicate that ADJP is erroneous (Bies et al., 1995, p. 120).

(2) a. [$_{UCP}$ federal/JJ ,/, state/NN and/CC local/JJ] public officials



b. [$_{ADJP}$ scientific/JJ ,/, engineering/NN and/CC academic/JJ] communities



It is not only identical daughters lists which must share the same mother, but also very similar daughters lists (Dickinson, 2006, 2009). If, for example, the daughters lists ADVP RB ADVP and ADVP , RB ADVP, as shown in (3), are treated as the same daughters list, then there are two different mothers, PP and ADVP (adverbial phrase), and this variation points to the presence of an error, in PP in this case.

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

b. to report their purchases and sales [$_{ADVP}$ [$_{ADVP}$ immediately] ,/, not/RB [$_{ADVP}$ a month later]]

In the next section, we define what it means for daughters lists to be similar enough to warrant having the same mother. A side effect of defining similarity is that some rules are similar to nothing else: these daughters lists are suspected not to be well-formed. For categories such as PP, where there is generally a clear head (IN, TO (*to*), or PP), one could search for rules without such a head (see, e.g., appendix A of Hockenmaier, 2003) to flag errors. By looking for rules which are not similar to others, however, we hope to find such types of cases automatically.

3.2 Basic valency inconsistencies

Equivalence criteria

Setting up equivalence classes between rules requires us to generally define when they are being used in the same way. For detecting inconsistencies in the labeling of a mother, we use the idea that anything not contributing to prediction of the rule's mother can be ignored. We use fairly simple properties to establish equivalences between rules. Namely, we employ the following steps for each daughters list:

1. Remove daughter categories that are always non-predictive to phrase categorization.
2. Group head-equivalent lexical categories.
3. Model adjacent identical elements as a single element.

The first step removes daughter categories that are always adjuncts and thus do not predict the mother phrase. For the WSJ data, there are a few cases to handle. First, there is punctuation, which is not involved in predicate-argument structure (cf. Hollingshead et al., 2005) and is often not included in treebank rules (cf. Brants et al., 2002).[3] Secondly, parentheticals (PRN) are always adjuncts when they occur in a daughters list. Finally, empty elements (i.e., -NONE-) can refer to any category, and thus they are uninformative with respect to the mother category.

The second step groups what we call *head-equivalent* lexical categories. Phrases headed by, for example, either singular common noun (NN) or plural common noun (NNS) are generally NP (noun phrase), and this distinction does not seem to add any information in predicting the mother. The full set of mappings is given in table 1, which is similar to mapping #2 in Hepple and van Genabith (2000). All other lexical tags, 13 of them, are only equivalent to themselves. It is important to note that the mappings here were originally motivated by a desire to predict the same mother; as we will discuss in section 3.4, these mappings are not entirely accurate at predicting the same syntactic distribution (e.g., NN and NNS behave differently with respect to determiners).

The final step models adjacent identical elements as a single element. This is akin to modeling a flat series of identical categories with the Kleene + operator (XP+). For instance, for the daughters list NN IN NP IN NP, the second IN NP says nothing more about predicting the

[3]The specific punctuation marks we remove are: ',', '.', ':', ';', '"', '(', ')', '$', and '#'. One could certainly revise this, especially as, for example, quotation marks sometimes indicate titles (cf. (8)), but initial experiments showed little difference in adjusting the set.

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

TABLE 1 Head-equivalence classes in the PTB

mother category, and so this is mapped to NN IN NP. All repetitious sequences are iteratively mapped into the shortest possible sequence that is still predictive. On each iteration, the longest possible identical adjacent sequence is mapped and passed to the next iteration; this continues until nothing more can be reduced. JJ NN NN JJ NN, for example, first reduces to JJ NN JJ NN, and then to JJ NN. This Kleene reduction step bears much in spirit to the correcting of illegal syntactic structures for the Penn Korean Treebank in Han et al. (2002), where legitimate right-hand sides of rules are hand-encoded as regular expressions, in order to detect and correct illegal syntactic structures. For predicting the correct mother, as we will see, this is a valid step, but for determining valency, we need to keep identical sequences intact.

Evaluation of equivalence criteria To evaluate whether or not equivalence classes lose any predictive information, we sampled 100 equivalence classes from section 00 of the WSJ and examined them by hand. In 98 cases, nothing predictive has been removed. In fact, in 24 of those cases, the mapping is complete, and nothing more could have been removed.

The two unsuccessful mappings involve the label NAC (not a constituent), as illustrated in (4). The daughters list NNP , NNP , reduces to NNP, but NNP by itself does not predict NAC. In fact, Bies et al. (1995, p. 208) mention that the presence of a comma partially determines NAC here, showing the affect the annotation scheme has on defining equivalence classes.[4]

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

In table 2, we present the average number of rule types and tokens per equivalence class, as a way to get a sense of what the practical effect of using equivalence classes is. We present this for each mother (LHS)

[4]We can note that many similar instances are not labeled NAC in the treebank.

category, as well as the overall numbers. In total, 15,246 rule types are mapped to 4382 classes (3.5 types per class), indicating that, with respect to predicting a mother category, there is much redundancy in a rule. Even without the last step of Kleene reduction, there are only 6419 equivalence classes (2.4 types per class), potentially indicating that for determining similarity among rules, a good deal of information can be ignored. Unsurprisingly, the most frequent categories (NP, S, VP) see the greatest rule type reductions, i.e., the most types per class.

		None		Step 1		Step 2		Step 3	
Category	Rules	Ty.	Tok.	Ty.	Tok.	Ty.	Tok.	Ty.	Tok.
ADJP	608	1.0	24.0	1.3	31.4	1.8	42.8	2.2	53.7
ADVP	298	1.0	75.3	1.1	85.0	1.4	107.4	1.8	136.9
CD	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CONJP	10	1.0	30.2	1.0	30.2	1.0	30.2	1.3	37.8
FRAG	239	1.0	2.2	1.6	3.5	1.6	3.5	1.8	3.8
INTJ	24	1.0	5.3	1.3	7.1	1.3	7.1	1.6	8.5
LST	8	1.0	4.8	2.0	9.5	2.0	9.5	2.0	9.5
NAC	47	1.0	8.7	1.2	10.3	1.3	11.1	1.7	15.2
NP	6282	1.0	55.8	1.4	75.2	2.3	125.6	4.2	234.6
NX	152	1.0	8.8	1.2	10.2	1.4	13.2	2.2	18.9
PP	334	1.0	286.2	1.5	434.5	1.6	464.0	1.9	543.1
PRN	253	1.0	9.6	2.4	22.8	2.6	24.4	3.0	28.5
PRT	9	1.0	293.4	1.0	293.4	1.5	440.2	1.5	440.2
PRT|ADVP	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
QP	347	1.0	27.2	1.2	33.0	1.4	38.1	1.9	50.8
RRC	16	1.0	2.9	1.1	3.1	1.1	3.1	1.1	3.1
S	1675	1.0	59.8	3.8	228.5	3.9	230.6	4.9	290.1
SBAR	178	1.0	171.5	1.7	285.4	1.7	293.6	1.9	328.3
SBARQ	57	1.0	3.9	2.1	7.9	2.0	7.9	2.0	7.9
SINV	211	1.0	9.7	1.6	15.8	1.9	18.7	2.2	21.3
SQ	117	1.0	3.0	1.3	3.9	1.8	5.3	1.8	5.5
UCP	192	1.0	2.6	1.2	3.2	1.3	3.4	1.5	3.9
VP	3967	1.0	36.9	1.8	64.4	3.5	130.1	4.9	179.1
WHADJP	10	1.0	5.9	1.0	5.9	1.0	5.9	1.0	5.9
WHADVP	22	1.0	119.9	1.1	131.9	1.1	131.9	1.2	146.6
WHNP	119	1.0	76.6	1.1	81.4	1.4	104.8	1.6	123.2
WHPP	5	1.0	78.4	1.3	98.0	1.3	98.0	1.3	98.0
X	64	1.0	2.8	1.2	3.3	1.3	3.5	1.4	3.8
Total	15,246	1.0	52.0	1.56	80.9	2.4	123.5	3.5	180.9

TABLE 2 Average number of rule types per equivalence class (*Ty.*) and rule tokens per equivalence class (*Tok.*) after each stage of rule reduction

The interpretation of equivalence classes We have described the mapping into equivalence classes with the goal of predicting the same mother category. However, these classes have a more linguistic interpretation, in reducing a rule to its essential elements. With the exception

of the third step—which potentially changes the set of dependents—using these classes is an attempt to map rules with the same valency to one another.

We can rephrase the intention of the first two steps as the following:

1. Remove non-linguistic material.
2. Group categories which have the same syntactic distribution.

This is a fairly crude way to ask whether two rules share the same valency. In the coming sections, we will try to refine this notion such that we better compare different rules with respect to their valents.

Dissimilarity

Viewing equivalences as a way to capture the essential linguistic elements of a rule has a noticeable consequence: a rule which does not have any equivalents in the grammar is more likely to be "non-linguistic," i.e., an error or other anomaly. Consider an example like (5): the daughters list RB TO JJ NNS has no similar rules in the treebank—i.e., this is a valency which no other rule shares—and it indeed is erroneous (Bies et al., 1995, p. 179). Detecting rules without equivalents is thus our first attempt at detecting ad hoc rules.

(5) [NP close/RB to/TO wholesale/JJ prices/NNS]

While anomalous valency detection can turn up ad hoc rules—confirmed by the evaluation in section 4.1—this relies on a very strict notion of similarity, when in practice rules are similar, though not equivalent. The method needs to be refined to ensure higher accuracy, more coverage, and more generality.

3.3 Ad hoc detection with equivalence classes

Backing off from strict (non-)equivalence, we explore two ways to define the similarity between rules, relying on comparisons for the entire rule (*whole daughters method*) and comparisons of local parts of a rule, namely bigrams (*bigram method*). Other options could be explored, but these provide different ends of a spectrum of rule abstractions.

As a first pass, we also employ equivalence classes since we presume that they are capturing linguistically-sensible properties of similarity. We refer to these scores as *reliability scores*. After mapping rules to equivalence classes, the general methods for determining similarity (described next) are then used to determine which of the equivalence classes is better or worse. In section 3.4, we discuss removing the dependence on equivalence classes and being corpus-independent.

Whole daughters method with equivalence classes

A simple way of calculating similarity to reflect changes in valency is to use edit distance between rules. This reflects the intuition that rules with similar lists of daughters reflect the same properties. We operationalize this intuition by assigning each rule type a reliability score as follows, where equivalence classes do not include Kleene reduction:

1. Map a rule to its equivalence class.
2. Add 1 for every rule token within the equivalence class.
3. Add $\frac{1}{2}$ for every rule token within a highly similar equivalence class.

This particular method of scoring is a "positive" way of scoring rules, in that in step #3 we look for support from other rules that the rule is legitimate. Rules without such evidence—i.e., with the lowest scores—are flagged as potentially ad hoc. There are of course many possible variants to scoring, e.g., different weightings for the second and third steps, but this definition contains all the essential elements. In section 3.4, we will to some extent tease apart the contributions of the different components as we improve upon the methods, showing that the similarity check in step #3 is the essential part of the process.

To determine similarity, we use a modified Levenshtein distance, where only insertions and deletions are allowed, and a distance of one between equivalence classes qualifies as *highly similar*.[5] As for the first modification, we do not utilize substitutions, as they are too problematic, given the difference in meaning of each category. Consider the problematic rule in (6), which occurs once. If we allow substitutions, then we will find 760 "comparable" instances of VP → VB, despite the vast difference in category (verb (VB) vs. adverb (RB)). As for the second modification, allowing two or more changes would allow us to add and subtract dissimilar categories.

(6) VP → RB

This definition of similarity captures generalizations such as adverbial phrases being optional. For example, in (7), after removing punctuation, the rule reduces to S → PP ADVP NP ADVP VP. With a strict notion of equivalence, there are no comparable rules. However, the class S → PP NP ADVP VP, with 198 members, is highly similar, indicating more confidence in this correct rule.

[5]The score is more generally $\frac{1}{1+distance}$, but we cut the distance at 1.

(7) [$_S$ [$_{PP}$ During his years in Chiriqui] ,/, [$_{ADVP}$ however] ,/, [$_{NP}$ Mr. Noriega] [$_{ADVP}$ also] [$_{VP}$ revealed himself as an officer as perverse as he was ingenious] ./.]

In determining similarity, it may seem that allowing any insertion or deletion is too lenient: rules could have multiple different kinds of arguments, and removing or adding one changes the argument structure. The method finds problematic cases with valencies like no other rule, as, for example, a verb with four NP sisters is hard to justify. Relatedly, we do not treat heads specially, as identifying them would require additional treebank-specific information; if one were to include this information—e.g., not allow heads to be deleted or inserted—then the method would likely be more effective.

Using this defintion of similarity, we can see the average number of similar rules (both types and tokens) per equivalence class in table 3. This is the number of rules which are similar, but not equivalent, to a given equivalence class. For example, for an average NP equivalence class, after step 2, there are 2.25 equivalent rule types within the class (table 2) and 5.78 similar rule types (table 3). Some of the averages go down (e.g., ADVP) because as the number of equivalent rules increases, the number of similar ones potentially decreases. In general—even without any equivalence class mappings (i.e., the *None* column)—similarity pulls in a large number of new rules to provide evidence for a given rule, indicating its potential utility without equivalence classes (section 3.4).

Bigram method with equivalence classes

The other method of detecting ad hoc rules abstracts a rule to its component parts, namely bigrams, including added *START* and *END* tags. Bigrams capture contextual information and can be abstracted from any rule. This abstraction is similar to features using information about n-grams of daughter nodes in parse reranking models (e.g., Collins and Koo, 2005). We assign reliability scores to rule types as follows:

1. Map a rule to its equivalence class, resulting in a *reduced rule*.
2. Calculate the frequency of each <mother,bigram> pair in a (reduced) rule: for every (reduced) rule token with the same pair, add a score of 1 for that bigram pair.
3. Assign the score of the least-frequent bigram as the score of the rule.

Because we are interested in anomalous sequences, we assign the score of the lowest-scoring bigram, essentially examining the weakest parts, in order to more directly look for "negative" evidence that a

Category	None		Step 1		Step 2		Step 3	
	Ty.	Tok.	Ty,	Tok.	Ty.	Tok.	Ty.	Tok.
ADJP	3.7	439.9	4.3	549.3	4.3	713.8	4.3	896.9
ADVP	3.8	1828.7	4.0	1991.6	4.2	2062.5	4.1	2568.7
CD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CONJP	2.0	78.0	2.0	78.0	2.0	78.0	2.5	121.0
FRAG	2.7	11.3	3.7	26.9	3.7	26.9	3.8	30.8
INTJ	1.8	12.2	1.9	21.6	2.0	22.3	1.8	21.9
LST	1.8	10.5	1.0	9.0	1.0	9.0	1.0	9.0
NAC	2.6	44.0	2.7	53.0	2.8	51.5	2.8	69.8
NP	5.1	1835.6	5.7	2252.2	5.78	3470.0	5.5	6665.4
NX	3.2	68.5	3.6	83.0	3.5	109.1	3.3	150.9
PP	3.3	8645.0	3.6	9438.6	3.7	9336.0	3.7	10,032.9
PRN	2.8	58.6	3.4	223.8	3.6	239.8	3.5	270.3
PRT	1.0	293.4	1.0	293.4	1.0	440.2	1.0	440.2
PRT|ADVP	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
QP	3.9	258.3	3.9	395.5	4.3	452.4	4.2	625.8
RRC	1.3	3.3	1.1	3.3	1.1	3.3	1.9	5.7
S	4.0	2200.9	4.8	8580.1	4.8	7996.2	4.7	9598.4
SBAR	3.2	2459.2	3.5	4344.5	3.5	4249.3	3.3	4640.7
SBARQ	2.4	19.3	2.4	43.3	2.5	43.4	2.5	43.4
SINV	2.6	86.9	3.2	148.0	3.5	178.0	3.6	206.6
SQ	2.5	14.6	2.8	24.2	2.9	39.2	3.3	43.9
UCP	1.9	7.6	2.3	10.8	2.2	11.5	2.3	12.8
VP	4.8	731.0	5.7	1153.9	5.9	3374.2	5.8	4643.1
WHADJP	1.6	15.7	1.6	15.7	1.6	15.7	1.6	15.7
WHADVP	2.7	747.6	2.9	793.8	2.9	793.8	2.7	772.8
WHNP	2.8	386.7	3.0	380.3	3.2	563.9	3.2	717.3
WHPP	1.0	78.4	1.0	97.8	1.0	97.8	1.0	97.8
X	1.7	5.8	2.3	8.6	2.3	9.6	2.6	11.6
Overall	4.5	1473.5	5.1	2085.7	5.0	3152.3	4.7	4605.0

TABLE 3 Average number of *similar* rule types (*Ty.*) & tokens (*Tok.*) per equivalence class, after each stage of rule reduction (not including counts for *equivalent* rules)

rule is ad hoc. This is in the spirit of Květon and Oliva (2002), who define invalid bigrams for POS annotation sequences in order to detect annotation errors.[6]

As one example, consider (8), where the reduced rule NP → NP DT NNP is composed of the bigrams START NP, NP DT, DT NNP, and NNP END. All of these are relatively common (more than a hundred occurrences each in the WSJ), except for NP DT, which appears in only two other rule types. Indeed, DT (determiner) is an incorrect tag (NNP [proper noun] is correct[7]): when NP is the first daughter of another NP,

[6]One could also explore a scoring method with bigrams which is more parallel to the whole daughters method, as in Dickinson (2010).

[7]It may also be the case that NP-TTL is applicable here (Bies et al., 1995, p. 49); regardless, there is some error here.

it is generally a possessive, precluding the use of a determiner.

(8) (NP (NP ABC 's) (`` '') (DT This) (NNP Week))

The whole daughters method misses such problematic structures because it does not explicitly look for local anomalies. The disadvantage of the bigram method, however, is its missing of the bigger picture: for example, the erroneous rule NP → NNP CC NP will get a large score because each subsequence is quite common. But this exact sequence is rather rare, since NNP and NP are not generally coordinated, so the whole daughters method will assign a low score.

These two methods, using equivalence classes, are evaluated in section 4.2.

3.4 Ad hoc detection without equivalence classes

As discussed in section 3.2, equivalence classes capture important linguistic facts, namely: categories which (sometimes) have the same syntactic distribution should be treated equivalently, and some categories are (sometimes) optional. Mapping each rule to a single equivalence class treats these facts as hard constraints, when in reality they are only true in certain contexts (discussed below). Removing a step of mapping rules into such classes will allow us to capture this "softer" nature of similar syntactic distributions and at the same time remove the methods' reliance on treebank-specific information.

A first issue with equivalence classes is that 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 sometimes are syntactic differences. As an example, consider (9): comparative adjectives are used in correlative *the*-clauses (Bies et al., 1995, p. 303ff), whereas positive adjectives generally are not. This is a syntactic context where the two categories are not replaceable; in this case, they need to be treated separately.

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

Secondly, consider the redundancy in using equivalence classes: by comparing rules to similar rules, we are already naturally capturing equivalences among rules. The fact that the categories JJ and JJR often are replaceable can be deduced by their similar behavior 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, we can work on a variety of corpora without learning each annotation scheme, and we can be more sensitive to the surrounding categories. For example, a category may always be an adjunct, but it is often important to know where the adjunct is. Consider (10a), with the rule ADVP → RB RB -LRB- PP -RRB- PP. Although both -LRB- and -RRB- (codes for left and right brackets) are adjuncts, observing them within another structure is odd, given that the preferred analysis is to embed such bracketed material, as in (10b).

(10) 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.]]

Although we now advocate abandoning equivalence classes, the equivalences still provide a useful goal. Levenshtein distance, for example, is effective as a means for determining similarity because it captures properties such as the removability of adjuncts and the distributional equivalence of categories.

Whole daughters method without equivalence classes

For both methods, the reliability scoring is based on two major components, frequency of a rule and its similarity to other rules. Removing the equivalence class restriction, we have the following as our method for computing the whole daughters score:

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

This is what is reported for the whole daughter *reliability* method in section 4.3 (see table 6). It is clear 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 want to determine the role of similarity in the detection of ad hoc rules. When we calculate *similarity* scores for the whole daughters method, we simply count one for each similar rule (instead of $\frac{1}{2}$), as this has a natural interpretation of the number of similar rules (see table 9).

Bigram method without equivalence classes

Likewise, for the bigram method, we can rely only on similarity scoring and factor out the frequency of the rule in question, in addition to trying reliability scores. The *similarity scores* are calculated as follows:

1. Calculate the frequency of each <mother,bigram> pair in a rule.
2. Assign the score of the least-frequent bigram as the score of the rule, *minus the frequency of the rule in question.*

Reliability scores are calculated in the same way, but without the subtraction in step #2.

These two methods, without relying on equivalence classes, are evaluated in section 4.3.

4 Evaluation of different methods

Having defined different ways to detect ad hoc rules, we now evaluate them. We evaluate on a context-free treebank, namely the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB, Marcus et al., 1993). In subsequent sections (5 and 6), we will discuss extending the methods to treebanks with discontinuous constituents.

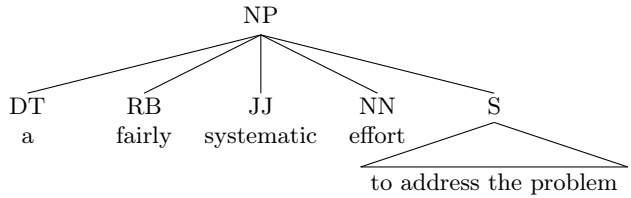
4.1 Evaluation of basic valency inconsistencies

For the first experiment, we examine the entire WSJ, following Dickinson and Meurers (2005b). As described in section 3.2, we group rules into equivalence classes (including Kleene reduction) and examine which rules have no equivalents, of which there are 2141 in the WSJ. From a random sample of 100 of these 2141 unique rules, we find that 39 of them are errors. Thus, we estimate 835 erroneous rules overall, with a 95% confidence interval of 630 to 1040 errors. Perhaps more telling is the nature of the identified rules, which we now turn to.

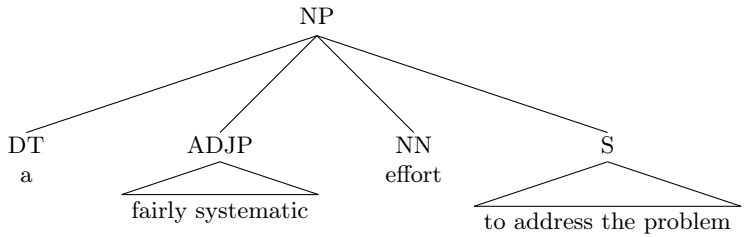
Errors One of the major sources of errors is the overapplication of flat structures in the treebank. In the WSJ, many constructions are supposed to have no internal structure, such as nominal modifiers which are themselves nouns, but many cases still require more structure. In (11), for example, the RB JJ sequence annotated as in (11a) should be bracketed as an ADJP, as in (11b). We find these cases because the RB JJ sequence is anomalous without the ADJP layer of structure.

(11) there seems * to be [$_{NP}$ a/DT fairly/**RB** systematic/**JJ** effort/NN [$_{S}$ * to address the problem]]

a.



b.



Likewise, our non-equivalence method finds categories which appear in the wrong context. In (12), for instance, the category DT appears in the daughters list DT ADJP CC PP, which should be CC ADJP CC PP (cf. Santorini, 1990, p. 9). Given the context of the whole rule (where CC is present), we are able to detect a mis-annotated category.

(12) [$_{UCP}$ both/**DT** [$_{ADJP}$ prudent] and/CC [$_{PP}$ in the best long-term interest of the shareholders]]

Ungrammatical constructions Four constructions in our sample cover ungrammatical language,[8] for example, the rule QP → RB JJ $ CD, illustrated in (13). This is the best analysis given the annotation scheme, but is clearly for ill-formed constructions (e.g., *as little $ 3* should be *as little as $ 3*). Likewise, as shown in (14), there are rules which are used to cover "financialspeak."

(13) . . . they 're charging [$_{QP}$ as/RB little/JJ $/$ 3/CD] *U* a day .

(14) [$_{NP}$ [$_{NP}$ Net income] [$_{X}$ *] :/: [$_{NP}$ $ 599.9 million *U* ; or $ 20.20 *U* a share]]

Annotation scheme & guidelines Perhaps most beneficial for treebanking purposes is the fact that identifying rules unlike any other uncovers annotation practices which are non-uniform. As one example, we find issues with the category QP (quantifier phrase), used for complex numerical determiners. Both *as little as* and ranges of dollar amounts

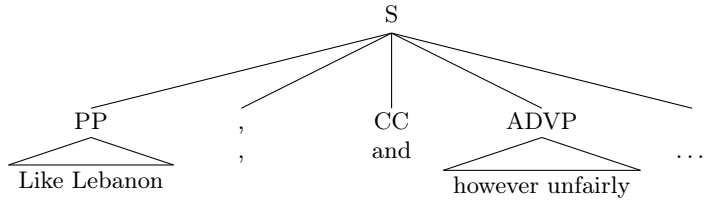
[8] While the term "ungrammatical" can be debatable, we use it in instances where the sentence is simply non-English; see, e.g., Foster (2007) for more discussion of grammaticality and the need to distinguish it in natural language parsing.

are supposed to be annotated as flat QPs (Bies et al., 1995, p. 194-202), but there is little guidance on what to do when they are together, and so we find flat QP structures, as in (15).

(15) [$_{QP}$ as/RB little/JJ as/IN $/$ 89/CD to/TO $/$ 109/CD]

As another example, consider a case like (16), which arguably contains a parenthetical, namely the string *and however unfairly*. Since parentheticals (PRN) are "determined ultimately by individual annotator intuition" (Bies et al., 1995, p. 50), we cannot say that this analysis is incorrect.

(16) [$_{S}$ [$_{PP}$ Like Lebanon] ,/, and/CC [$_{ADVP}$ however unfairly] ,/, [$_{NP}$ Israel] [$_{VP}$ is regarded * by the Arab world as a colonial aberration] ./.]



4.2 Evaluation of ad hoc detection with equivalence classes

We now turn to the detection of ad hoc rules with a more general notion of similarity (either whole daughters or bigrams), using equivalence classes as a first step (section 3.3). With a more fine-grained definition of similarity, we are now in a position not only to examine rule quality, but also rule *utility*, i.e., whether rules are needed for new data. Thus, to gauge our success in detecting ad hoc rules, we evaluate the reliability scores in two main ways: 1) whether unreliable rules generalize to new data and 2) whether the unreliable rules which do generalize are ad hoc in other ways—e.g., erroneous. To measure this, we use sections 02-21 of the WSJ corpus as training data to derive scores and section 24 as evaluation data for development. This evaluation will help to understand the strengths and weaknesses of the two methods. We reserve section 23 as test data for the final methods in section 4.3, where we also perform n-fold cross-validation.

Ungeneralizable rules

In this section and in section 4.3, we examine how many rules from the training data do not appear in the evaluation data, for different thresholds, referring to this as the *ungeneralizability rate*. In table 4, for example, the method identifies 3548 rules with scores less than or equal to 50, 3439 of which do not appear in the evaluation data, resulting in

an ungeneralizability rate of 96.9%. This measures the degree to which a rule was only needed for one particular data set.

To interpret the tables below, we first need to know that of the 15,246 rules from the training data, 1832 occur in the evaluation data, or only 12.02%, corresponding to 27,038 rule tokens. There are also 396 new rules in the evaluation data, making for a total of 2228 rule types and 27,455 rule tokens. The percentage of 12% appears quite low, illustrating the need for doing this work in the first place: most training rules are irrelevant for analyzing this new text.[9]

The results are shown in table 4 for the whole daughters method and in table 5 for the bigram method. Both methods successfully identify rules with little chance of occurring in new data, the whole daughters method performing slightly better.[10]

Threshold	Rules	Unused	Ungeneralizability
1.0	311	311	100%
25.0	2683	2616	97.5%
50.0	3548	3439	96.9%
100.0	4596	4419	96.2%

TABLE 4 Ungeneralizability of whole daughters method, using equivalence classes (WSJ-24)

Threshold	Rules	Unused	Ungeneralizability
1	599	592	98.8%
5	1661	1628	98.0%
10	2349	2289	97.4%
15	2749	2657	96.7%
20	3120	2997	96.1%

TABLE 5 Ungeneralizability of bigram method, using equivalence classes (WSJ-24)

We have isolated thousands of rules with little chance of being observed in the evaluation data, and, as we will see in the next section, many of the rules are problematic in other ways. The reliability scores

[9] Treebanks constructed with pre-existing grammars may display a different pattern, and in general one could use such measurements to investigate the quality and portability of linguistic annotation.

[10] For additional results with this method, orthogonal to the main discussion here, see Dickinson (2008).

use token counts, but have the advantage of being able to identify infrequent but correct rules (cf. example (7)) and frequent but unhelpful rules. For example, in (17), we find erroneous cases from the evaluation data of the rules WHNP → WHNP WHPP (the node dominating *five* should be NP) and VP → NNP NP (*OKing* should be VBG). These rules appear 27 and 16 times, respectively, in the training data, but have scores of only 28.0 and 30.5, showing their unreliability.

(17) a. [$_{WHNP}$ [$_{WHNP}$ five] [$_{WHPP}$ of whom]]
b. received hefty sums for * [$_{VP}$ OKing/NNP [$_{NP}$ the purchase of ...]]

Other ad hoc rules

To figure out why some rules appeared in the evaluation data at all, we hand-examined the rules appearing in the development data. We report analysis for thresholds that resulted in approximately 100 flagged rules. For the whole daughters scoring, at the 50 threshold, 55 (50.9%) of the 108 rules in the development data are errors. Adding these to the ungeneralizable rules, 98.5% (3494/3548) of the 3548 rules are unhelpful for parsing, at least for this data set. An additional 12 rules cover non-English or fragmented constructions, making for 67 clearly ad hoc rules.

For the bigram scoring, at the 20 threshold, 67 (54.5%) of the 123 rules in the development data are erroneous, and 8 more are ungrammatical. This means that 97.9% (3054/3120) of the rules at this threshold are unhelpful for parsing this data.

Problematic cases As for the remaining rules which are not erroneous or ungrammatical, there are several cases which reveal non-uniformity in the annotation scheme or guidelines. Consider the case of NAC (not a constituent), used for complex NP premodifiers. The description for tagging titles in the guidelines (Bies et al., 1995, p. 208-209) covers the exact case found in section 24, shown in (18a). This correct rule, NAC → NP PP, is one of the lowest-scoring rules which occurs, with a whole daughters score of 2.5 and a bigram score of 3. Examining the guidelines more closely, however, we find examples such as (18b). Here, no extra NP layer is added, and it is not immediately clear what the criteria are for having an intermediate NP.

(18) a. a " [$_{NAC}$ [$_{NP}$ Points] [$_{PP}$ of Light]] " foundation
b. The Wall Street Journal " [$_{NAC}$ American Way [$_{PP}$ of Buying]] " Survey

Secondly, rules with mothers which are rare tend to always receive lower scores. For example, the rules dominated by SINV, SQ, or SBARQ

are all correct (6 in whole daughters, 5 in bigram), but questions are not very frequent in this news text: SQ appears only 350 times and SBARQ 222 times in the training data. One might thus consider normalizing the scores based on the frequency of the parent.

Finally, there are issues with coordinate structures, for both methods. For example, NP → NN CC DT receives a low whole daughters score of 7.0, despite the fact that NP → NN and NP → DT are very common rules. For the whole daughters method, of the 108 rules, 28 of them had a conjunct (CC or CONJP) in the daughters list, and 18 of these were correct. Likewise, for the bigram method, 18 had a conjunct, and 12 were correct. Reworking reliability scores to reflect coordinate structures and handle each case separately may require treebank-specific knowledge: the Penn Treebank, for instance, distinguishes unlike coordinated phrases (UCP) from other coordinated phrases.

Comparing the methods

In general, both methods fare badly with clausal rules (those dominated by S, SBAR, SINV, SQ, or SBARQ), but the effect is slightly greater on the bigram scoring, where 20 of the 123 rules are clausal, and 16 of these are correct (i.e., 80% are misclassified). This stems from the fact that most modifiers are adjoined at the sentence level when there is any doubt about their attachment (Bies et al., 1995, p. 13), leading to correct but rare subsequences. In sentence (19), for example, the reduced rule S → SBAR PP NP VP arises because both the introductory SBAR and the PP are at the same level, leading to a rare SBAR PP sequence and a bigram score of 13.

(19) [$_S$ [$_{SBAR}$ As the best opportunities for corporate restructurings are exhausted * of course] ,/, [$_{PP}$ at some point] [$_{NP}$ the market] [$_{VP}$ will start * to reject them] ./.]

The whole daughters method, on the other hand, assigns this rule a high reliability score of 2775.0, due to the fact that both SBAR NP VP and PP NP VP sequences are common. For some rules with long modifier sequences, the whole daughters method can be effective in a flat treebank since modifiers are easily skipped over in comparing to other rules.

The whole daughters method has difficulty with categories which have a highly varied set of possible heads and arguments, such as quantifier phrases (QPs). QP is "used for multiword numerical expressions ... where the QP corresponds frequently to some kind of complex determiner phrase" (Bies et al., 1995, p. 193). This definition leads to

rules which look different from QP to QP. Some of the lowest-scoring correct rules are shown in (20). We can see that there is not a great deal of commonality about what comprises quantifier phrases, even if subparts are common and thus not flagged by the bigram method.

(20) a. [$_{QP}$ only/RB three/CD of/IN the/DT nine/CD] justices
b. [$_{QP}$ nearly/RB twice/RB] the national average
c. 10 % [$_{QP}$ or/CC more/JJR]

4.3 Evaluation of ad hoc detection without equivalence classes

We now turn to the evaluation for ad hoc rule detection using general similarity metrics but without employing equivalence classes (section 3.4). First, we evaluate the reliability scores, and in the next section we look at the contribution of similarity alone. For an in-depth analysis of the different aspects of the method, we use the development data, i.e., WSJ-24, and in section 4.3, we confirm our results on the test data (WSJ-23). Table 6 shows the ungeneralizability rate for the whole daughters method, for different thresholds, and table 7 for the bigram method.

Threshold	Rules	Unused	Ungeneralizability
1.0	1625	1617	99.5%
2.0	2801	2785	99.4%
3.0	3515	3479	99.0%
4.0	4011	3965	98.9%
5.0	4412	4357	98.8%

TABLE 6 Ungeneralizability of whole daughters method (reliability scores), without equivalence classes (WSJ-24)

The results are quite good and dramatically surpass the values presented in the previous section (cf. tables 4 and 5). For example, a threshold of 50 in the previous whole daughters method with equivalence classes identifies 3548 rules with 96.9% ungeneralizability. For that same approximate number of rules (threshold=3.0), the method without equivalences has 99.0% precision. The bigram method shows similar improvement. These results indicate that the methods can perform well without recourse to equivalence classes.

Equivalence classes Examining some cases by hand, we can analyze how the method changes by removing the equivalence class criteria. As an example affecting both methods, consider (21), with the rule NP → NP -LRB- NP , NP -RRB-. The whole daughters score goes from

Threshold	Rules	Unused	Ungeneralizability
1	1157	1143	98.8%
2	1857	1828	98.4%
3	2341	2296	98.1%
4	2721	2665	97.9%
5	3054	2985	97.7%

TABLE 7 Ungeneralizability of bigram method (reliability scores), without equivalence classes (WSJ-24)

5232.0 to 1.0, and the bigram method likewise goes from 12,368 to 40. The equivalence mappings reduce this rule to NP → NP NP NP, thus losing crucial information about how bracketing should be done for parenthetical information.

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

There are cases, on the other hand, in which not using equivalence classes is worse, assigning low scores to correct rules. For instance, in example (22), 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. An important question for working with an annotation scheme, then, is to determine which elements are or are not informative.

(22) [$_{S}$ -LRB-/-LRB- “/“ [$_{NP}$ Quest for Fire] ”/” [$_{VP}$ was the first time] ./. -RRB-/-RRB-]

The effect of similarity

One question about rule ungeneralizability is: to what extent is this an effect of frequency and to what extent an effect of similarity (cf. section 3.4)? As we can see in table 8, 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 useful rules from infrequent problematic rules.

For the similarity scores (i.e., number of similar rules without including the frequency of the rule in question), table 9 shows that the whole daughters method is effective at identifying ungeneralizable rules. More than that, this scoring is providing more fine-grained information.

As shown in table 10, the bigram method is less accurate in terms of generalizability, with a 94.5% rate for scores of 0, for example.

Threshold	Rules	Unused	Ungeneralizability
1	8776	8627	98.3%
2	10,741	10,475	97.5%
3	11,601	11,253	97.0%
4	12,131	11,723	96.6%

TABLE 8 Ungeneralizability results based on rule frequency, without equivalence classes and without using similarity (WSJ-24)

Threshold	Rules	Unused	Ungeneralizability
0	1851	1819	98.3%
1	2622	2571	98.1%
2	3147	3080	97.9%
3	3538	3454	97.6%
4	3865	3769	97.5%
5	4149	4041	97.4%
85,957	15,246	13,414	88.0%

TABLE 9 Ungeneralizability of whole daughters method (similarity scores), without equivalence classes (WSJ-24)

While this might indicate worse prediction, a cursory hand-examination also shows that many of the identified rules are structures that seem to reflect errors or non-uniform annotation scheme decisions, such as CONJP → IN RB.

Threshold	Rules	Unused	Ungeneralizability
0	1625	1535	94.5%
1	2323	2183	94.0%
2	2801	2631	93.9%
3	3178	2986	94.0%
4	3494	3284	94.0%
5	3781	3548	93.8%
33,907	15,246	13,414	88.0%

TABLE 10 Ungeneralizability of bigram method (similarity scores), without equivalence classes (WSJ-24)

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 sampled 100 WSJ rules occurring only once in the training data. Without examining the rule scores, we marked each as an error, ungrammatical, unclear, or correct. Of these 100, 21 are errors, and 5 cover ungrammatical

constructions.

For the whole daughters method, we find that the bottom 22 cases (scores = 0) contain 8 errors, as well as 3 ungrammatical structures. For the bigram method, the bottom 26 cases (scores ≤ 5) have 8 errors and 3 ungrammatical constructions. By finding 38% of the errors and 60% of the ungrammatical cases with just the bottom quarter of cases, these similarity-based scores are, in practice, effective at sorting problematic low-frequency rules from less problematic ones.

Results on test data

We confirm that the methods identify less useful rules on the test data, using similarity alone. The whole daughters results are given in table 11, while the bigram results are in table 12. The results are comparable to those for the development data (compare tables 9 and 10).[11]

Threshold	Rules	Unused	Ungeneralizability
0	1851	1818	98.2%
1	2622	2569	98.0%
2	3147	3064	97.4%
3	3538	3437	97.1%
4	3865	3747	96.9%
5	4149	4009	96.6%
85,837	15,246	12,980	85.1%

TABLE 11 Ungeneralizability of whole daughters method (similarity scores), without equivalence classes (WSJ-23)

Threshold	Rules	Unused	Ungeneralizability
0	1625	1519	93.5%
1	2323	2152	92.6%
2	2801	2588	92.4%
3	3178	2935	92.4%
4	3494	3228	92.4%
5	3781	3491	92.3%
33,907	15,246	12,980	85.1%

TABLE 12 Ungeneralizability of bigram method (similarity scores), without equivalence classes (WSJ-23)

The bigram method consistently shows lower performance than the whole daughters method. There are still a few reasons, however, to in-

[11] For evaluation showing similar behavior across different genres, see Dickinson and Foster (2009).

coporate the bigram method into a search for ad hoc rules. First, as discussed before, the method complements the whole daughters method, in searching for different types of information. Secondly, as demonstrated earlier, both methods successfully turn up other types of ad hoc rules, most notably errors. Finally, it is not clear which type of method would be better to incorporate into, for example, parser error detection work. Indeed, in Dickinson (2010), the bigram method sometimes outperforms the whole daughter method for parser error detection in different corpora.

N-fold cross validation

To further confirm the validity of flagging ad hoc rules with our two methods, we performed n-fold cross validation, taking three sections of the treebank at a time as the test data and all the rest as the basis for the training grammar. As there are a total of 25 sections (00-24), we do not include section 24 in any test set of our 8-fold cross-validation, though it is included in training sets. We report the average scores across the eight training scenarios in table 13 for the whole daughters method and table 14 for the bigram method.

Threshold	Rules	Unused	Ungeneralizability
0	13,347	12,694	95.1%
1	18,936	17,977	94.9%
2	22,613	21,358	94.5%
3	25,313	23,791	94.0%
4	27,642	25,879	93.6%
5	29,606	27,642	93.4%
All	108,572	84,805	78.1%

TABLE 13 Average ungeneralizability of whole daughters method (similarity scores), without equivalence classes (8-fold cross-validation)

The first thing to note is that, because three sections are used for test data, instead of just one, a lower percentage of rules are ungeneralizable (78.1% vs. 85.1% in section 23, as seen in tables 11 and 12). Given this, however, using low whole daughters scores is quite effective at finding ungenerablizable rules. The bigram method is less effective, though still an improvement over nothing at all.

5 Discontinuous constituents

In developing a corpus-independent method, we have focused to this point only on treebanks with context-free rules. For languages with relatively free constituent order, such as German, Dutch, or the Slavic

Threshold	Rules	Unused	Ungeneralizability
0	12,339	10,844	87.9%
1	17,329	15,165	87.5%
2	20,980	18,359	87.5%
3	23,812	20,779	87.3%
4	26,074	22,706	87.1%
5	27,814	24,130	86.8%
All	108,572	84,805	78.1%

TABLE 14 Average ungeneralizability of bigram method (similarity scores), without equivalence classes (8-fold cross-validation)

languages, the combinatorial potential of the language encoded in constituency cannot be mapped straightforwardly onto the word order possibilities of those languages. As a consequence, the treebanks that have been created, for example, for German (NEGRA, Skut et al., 1997; VERBMOBIL, Hinrichs et al., 2000; TIGER, Brants et al., 2002) have relaxed the requirement that constituents have to be contiguous. Allowing crossing branches makes it possible to syntactically annotate the language data without requiring postulation of empty elements as placeholders or making other changes to the data.

Discontinuous constituents are strings of words which are not necessarily contiguous, yet form a single constituent with a single label, such as the noun phrase *Ein Mann der lacht* in the example of German relative clause extraposition in (23) (Brants et al., 2002).[12]

(23) **Ein** **Mann** kommt , **der** **lacht**
a man comes , who laughs
A man who laughs comes.

Discontinuities pose a problem for comparing rules across a grammar, since the current methods tacitly assume that constituents in a rule are adjacent and non-overlapping. This issue is non-trivial: in the TIGER treebank, for example, 27.5% of graphs contain crossing edges (Hockenmaier, 2006).

5.1 An appropriate representation

Previous representations

Representing discontinuous rules is not a new issue; however, previous representations have focused more on issues of theoretical compactness

[12]The ordinary way of marking a constituent with brackets is inadequate for discontinuous constituents, so we instead boldface and underline the words belonging to a discontinuous constituent.

or parsing than on the comparison of rules. For example, ID/LP representations (see, e.g., Gazdar et al., 1985, Daniels and Meurers, 2004, Volk, 1996) separate ID (immediate dominance) relations from LP (linear precedence) relations. However, ID/LP constraints are designed to capture possibilities of rules, not their specific realizations.

Methods which convert discontinuous treebank structures into CFG trees are useful for parsing, but lose information. The "raising" conversion method lifts non-head nodes out of discontinuities (Kübler, 2005, Kübler et al., 2006). Consider figure 1 (from Boyd, 2007) for the sentence in (24). For the discontinuous VP (verb phrase), the category PP (prepositional phrase) is raised out of the VP, leading to rules of S → PP VMFIN VP and VP → VP VAINF. This method is clearly not appropriate for comparing local rules across a treebank, as it changes the rules.

(24) Mit dem Bau soll 1997 begonnen werden .
With the construction should 1997 begun be
Construction should start in 1997.

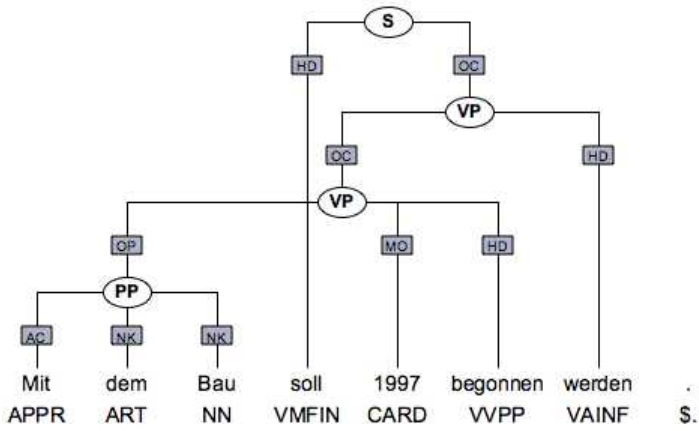


FIGURE 1 A sentence with crossing branches in TIGER

The "splitting" approach (Boyd, 2007) splits categories into contiguou spans, so as to create a context-free tree. For figure 1, for example, the rule for S (sentence) becomes S → VP* VMFIN VP*, representing the two parts of the rule. This leads to the rules VP* → PP and VP* → VP* VAINF, among other extra rules beyond what is in

the original tree, obscuring rules such as the lower VP rule with its PP, CARD (cardinal number), and VVPP (perfect participle) categories.

Reading rules off the trees

The representation we want needs to capture valency. As we have defined it, valency lists encode linear precedence (in addition to dominance relations), and our methods for ad hoc rule detection rely on this ordering. If we wish to lose no information in the rules, the problem with discontinuities is that of establishing linear precedence.

If there are intervening words between complete categories in a rule, then linear precedence within a rule can easily be established. For example, the rule NP → D N captures both dominance and precedence, as long as the category D completely precedes N—i.e., every item in the yield of D precedes every item in the yield of N—regardless of whether there are intervening words or whether D or N are themselves discontinuous. However, if N wraps around D, then both NP → D N and NP → N D lose information, as part of N is before D, and part of N is after it.

Discontinuous (-d) representation To directly encode discontinuities in a representation, we take all continuous subparts of a discontinuous daughter, and if there is more than one subpart, mark each one with an indicator (*-d*) that the category is discontinuous. Consider the example in figure 1, where there is a discontinuous VP. As all elements maintain strict orderings, the VP rules are straightforward: VP → PP CARD VVPP and VP → VP VAINF. The sentential (S) rule, however, represents the fact that the VP wraps around VMFIN (modal finite verb): S → VP-d VMFIN VP-d.[13]

This is much in the spirit of a splitting method (Boyd, 2007), but note how it differs: when there is a rule such as S → VP-d VMFIN VP-d, the representation in Boyd (2007) posits two separate VP-d (or VP*) rules for the daughter categories, in order to reconstruct a full tree. In order to maintain a representation which directly captures the valency in the treebank, we have only one rule for the VP daughter.

This type of representation is well-suited for detecting anamolous valencies: on the one hand, when there are strict orderings among elements, no additional notation is used, thereby representing all dependents of a head. On the other hand, discontinuities are often only licensed in certain situations. Thus, marking categories as split allows

[13]One could write the *d* on the side where there is a discontinuity, e.g., VP-d VMFIN d-VP, but we did not, due to a concern over data sparseness, especially considering categories such as d-VP-d; one could explore this further in the future, though our results are fairly strong as is (section 6).

one to capture these licensing contexts—e.g., the initial field before a finite verb, as in the S rule above.

We should note that, because we do not explicitly link different parts of a discontinuous category, sometimes information is lost in this representation. For example, in the rule S → NP-d VVFIN CARD NP-d NP-d PTKVZ NP-d, we cannot be sure which NP-d units are linked with one another. An alternative to this encoding is to index the discontinuities, as in S → NP-d$_1$ VVFIN CARD NP-d$_1$ NP-d$_2$ PTKVZ NP-d$_2$. While unambiguous, we do not wish to subdivide categories in such a way as to make category comparisons more difficult.

Flat representation Recall that the guiding principle for comparing rules across a treebank is to capture valency requirements and find those which are odd. An alternative approach to the explicit discontinuous representation above is to simplify and only capture the category information of a rule, ignoring discontinuities.

To achieve this, we start with the formalism of Discontinuous Phrase Structure Grammar (DPSG) (Bunt, 1996, Plaehn, 2004). In this formalism, daughters are listed as in CFG rules, but so are *context daughters*, indicating the material found within a discontinuous structure, but not a part of it. For the tree in figure 1, the rules would be as in (25), and in (25a), we find the inclusion of a context daughter in brackets: VMFIN is not a daughter of this VP, but occurs between its daughters. Additionally, using a notion of adjacency sequences, elements within a rule are ordered by the position of each's leftmost daughter: a constituent's leftmost daughter is required to precede the next constituent's leftmost daughter. In (25c), for example, defining precedence in terms of the leftmost daughter results in VP preceding VMFIN, even though in reality it, in some sense, wraps around the VMFIN.

(25) a. VP → PP [VMFIN] CARD VVPP
b. VP → VP VAINF
c. S → VP VMFIN

We use this *flat* representation, but remove the context daughter information, as context daughters do not indicate information about valency. Specifying precedence based on the leftmost daughter is a somewhat arbitrary decision, but the benefit is that we have a single VP daughter of S. One could explore using the rightmost daughters to determine precedence, but we expect little difference.

Function labels

As can been seen from figure 1, the TIGER treebank contains function labels, which have been ignored up to this point. However, work on

parsing German often includes such labels in their parsing and evaluation, as the labels indicate important properties regarding grammatical functions (see, e.g., Kübler et al., 2006). Thus, it seems worthwhile to explore the generalizability when function labels are and are not included. To do this, we simply expand each child label to include the function label. With a flat structure representation, for example, we will obtain the rules from figure 1 as shown in (26).[14] Note that we do not include the function label in the mother, capturing the fact that the mother category is endocentrically determined, i.e., its function in the sentence is generally not determined by its children.

(26) a. S → VP-OC VMFIN-HD
b. VP → VP-OC VAINF-HD
c. VP → PP-OP CARD-MO VVPP-HD

Secondary edges

For the TIGER treebank, secondary edges are used for coordinate constructions which involve constructions such as gapping and right node raising (Brants et al., 2002). This means that a particular node may have both a primary parent and a secondary parent, as illustrated in figure 2 for the sentence in (27).[15]

(27) Die Inflation liegt bei 13 Prozent , die Industrieproduktion
The inflation lies at 13 percent , the industrial production
ist 1991 gesunken , die Zahl der Arbeitslosen gestiegen .
is 1991 decreased , the number of the unemployed increased .
Inflation is at 13 percent, the industrial output decreased in 1991, the unemployment numbers increased.

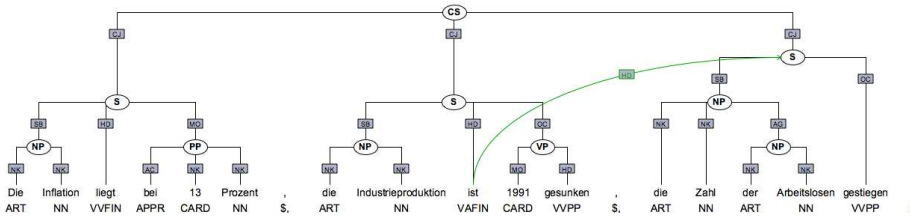


FIGURE 2 A graph with a secondary edge in TIGER

[14] OC=clausal object, HD=head, OP=prepositional object, MO=modifier

[15] Although the secondary edge in this tree is to the left, they can in principle occur in different locations.

On the one hand, these secondary edges seem to contain useful information about the parent—e.g., a rule like S → NP VVP becomes S → VAFIN NP VVP when the secondary information is included, i.e., actually containing a finite verb (VAFIN). On the other hand, if we treat secondary edges completely as we treat primary edges, at some point in a connected graph, two sisters will share the same descendent node, since the node has two parents. To obtain the informative value of secondary edges while avoiding overly cumbersome rules, we incorporate secondary edges only at the level at which they are introduced. For example, the secondary child of the S is included in the rule S → VAFIN NP VVP, but the CS (coordinated sentence) rule dominating the S does not include the secondary edge, thereby making this S child fully contiguous within the CS rule (CS → S S S).

6 Evaluation for discontinuous constituents

Data preparation To test the method on a corpus with discontinuous constituents, we use the TIGER corpus of German (Brants et al., 2002). For this, we split the data into training, development, and test sections, following the split in Dubey (2004). We include rules for ROOT nodes; the results obtained below are parallel when examining only non-ROOT rules.

Based on the previous evaluation (section 4), we only use *similarity* scores of our two methods, and we find similarity *without equivalence classes*; this is held constant across all tables of results in this section. Still, there are two different ways of breaking down the data (section 5.1): firstly, there is the issue of whether to explicitly represent discontinuities, or to use a completely flat structure. Secondly, there is the issue of whether to include function labels or not. Thus, we will present four sets of results on the two different methods.

Without function labels We start the examination for representations without function labels. For the whole daughters method, comparing the lowest thresholds in table 15 (flat representation) to those in table 16 (discontinuous representation), we can make an immediate observation: the lowest scores identify more rules for the representation explicitly marking discontinuities—e.g., 4814 vs. 3083 for a threshold of 0. With this extra notation, there are simply more rules. Likewise, the flat representation is more fine-grained, identifying fewer rules with each threshold. The results are strikingly on a par with each other, indicating that the similarity method is more important than the representation of discontinuous material.

Looking at tables 17 and 18, the results are slightly clearer for the

Threshold	Rules	Unused	Ungeneralizability
0	3083	3034	98.4%
1	4558	4477	98.2%
2	5706	5593	98.0%
3	6630	6484	97.8%
4	7389	7215	97.7%
5	8061	7847	97.3%
43,705	29,163	25,983	89.1%

TABLE 15 Whole daughter ungenerablizability with a flat representation, without function labels (development data)

Threshold	Rules	Unused	Ungeneralizability
0	4814	4718	98.0%
1	6768	6616	97.8%
2	8151	7947	97.5%
3	9234	8982	97.3%
4	10,130	9831	97.1%
5	10,904	10,558	96.8%
43,602	31,681	28,392	89.6%

TABLE 16 Whole daughter ungenerablizability with a discontinuous representation, without function labels (development data)

bigram method, where we observe a better ungeneralizability rate when using explicit discontinuities. We conjecture, however, that the bigram method works marginally better with explicit discontinuities because many local sequences involving the placement of discontinuous material are rarely found in new data.

Threshold	Rules	Unused	Ungeneralizability
0	1478	1388	93.9%
2	2896	2716	93.8%
4	3785	3548	93.7%
6	4640	4359	93.9%
8	5307	4975	93.7%
10	5902	5530	93.7%
16,742	29,163	25,983	89.1%

TABLE 17 Bigram ungenerablizability with a flat representation, without function labels (development data)

Threshold	Rules	Unused	Ungeneralizability
0	1794	1694	94.4%
2	3481	3284	94.3%
4	4593	4331	94.3%
6	5585	5264	94.3%
8	6347	5975	94.1%
10	7068	6652	94.1%
19,541	31,681	28,392	89.6%

TABLE 18 Bigram ungenerablizability with a discontinuous representation, without function labels (development data)

In addition to ungeneralizable rules, these methods turn up phrasal types which could have been assigned more intenal structure in another annotation scheme. For example, as the TIGER scheme set out to "annotate only the common minimum" (Brants et al., 2002), prepositional phrases have a variety of dependents, and we find cases such as PP → APPR PIAT ADJA ADJA NN PROAV (where APPR is the preposition).

With function labels The results for the same data, but with function labels included, are presented in tables 19-22. For the whole daughters method, we observe essentially the same numbers, while for the bigram method, ungeneralizability scores go from around 94% to 96%. This can partly be attributed to a higher baseline: adding function labels makes each rule rarer, resulting in a higher overall number of rules and a higher overall ungeneralizability rate. It seems that the bigram method benefits more by including more information in a rule: more specific categories lead to more specific bigrams and thus a greater chance of detecting an anomalous sequence.

Threshold	Rules	Unused	Ungeneralizability
0	4907	4829	98.4%
1	7112	6982	98.2%
2	8767	8592	98.0%
3	10,024	9797	97.7%
4	11,129	10,863	97.6%
5	12,013	11,702	97.4%
43,664	36,571	33,170	90.7%

TABLE 19 Whole daughter ungenerablizability with a flat representation, using function labels (development data)

Threshold	Rules	Unused	Ungeneralizability
0	6979	6852	98.2%
1	9633	9430	97.9%
2	11,567	11,297	97.7%
3	13,009	12,668	97.4%
4	14,181	13,784	97.2%
5	15,155	14,703	97.0%
43,561	39,105	35,608	91.1%

TABLE 20 Whole daughter ungenerablizability with a discontinuous representation, using function labels (development data)

Threshold	Rules	Unused	Ungeneralizability
0	3896	3760	96.5%
2	6967	6695	96.1%
4	8988	8628	96.0%
6	10,594	10,161	95.9%
8	11,715	11,218	95.8%
10	12,723	12,155	95.5%
16,742	36,571	33,170	90.7%

TABLE 21 Bigram ungenerablizability with a flat representation, using function labels (development data)

Threshold	Rules	Unused	Ungeneralizability
0	4501	4347	96.6%
2	7927	7628	96.2%
4	10,180	9775	96.0%
6	11,989	11,509	96.0%
8	13,242	12,697	95.9%
10	14,433	13,812	95.7%
19,541	39,105	35,608	91.1%

TABLE 22 Bigram ungenerablizability with a discontinuous representation, using function labels (development data)

Test data

To confirm our results, we select one group of settings from the development data to work with the TIGER corpus. Using function labels clearly led to higher ungeneralizability rates, but it is less clear whether the flat representation or the discontinuous representation is better, as the rates are nearly identical for similar numbers of identified rules.

Looking at the final row of tables 21 and 22, however, we see that the baseline for the flat representation is lower, meaning that a higher percentage of rules are used. Identifying unused rules at the same rate is actually better for the flat method, given that there is a greater overall chance of a rule being used. Thus, we report results on the test data for a flat representation with function labels, for both the whole daughters and bigram methods, as in tables 23 and 24. These tables show that the results hold for the test data.

Threshold	Rules	Unused	Ungeneralizability
0	4907	4845	98.7%
1	7112	7004	98.5%
2	8767	8605	98.2%
3	10,024	9811	97.9%
4	11,129	10,873	97.7%
5	12,013	11,714	97.5%
43,664	36,571	33,158	90.7%

TABLE 23 Whole daughter ungenerablizability with a flat representation, using function labels (test data)

Threshold	Rules	Unused	Ungeneralizability
0	3896	3778	97.0%
2	6967	6722	96.5%
4	8988	8656	96.3%
6	10,594	10,164	95.9%
8	11,715	11,216	95.7%
10	12,723	12,162	95.6%
16,742	36,571	33,158	90.7%

TABLE 24 Bigram ungenerablizability with a flat representation, using function labels (test data)

7 Related work

Work on flagging anomalous rules in treebanks has focused mainly on detecting errors in treebanks; a broad overview of this work can be found in chapter 1 of Dickinson (2005). Ule and Simov (2004) provide one example of work on detecting anomalous tree structures. To find the most unexpected tree nodes, Ule and Simov (2004) use information about the type of a node, the (grand)parent of that node, and the

children. An event which is unexpected, based on the χ^2 metric, is deemed likely to be an error. Ule and Simov are somewhat successful in finding errors and other anomalies in a treebank of 580 sentences: of the first 27 error candidates in a hand-checked test corpus, 11 were errors and five were the result of unclear guidelines. The main idea for their work is similar to the present work; a main difference lies in how one defines an unexpected event. For example, they do not compare daughters lists as we do.

For part-of-speech annotation, there is also work on anomaly detection. Eskin (2000) discusses how to use a sparse Markov transducer as a method for anomaly detection. In this context, an anomaly refers to a rare local tag pattern, found by using a mixture model to detect outliers. The method flags 7055 anomalies for the Penn Treebank, about 44% of which hand inspection shows to be errors. Similar to Eskin (2000), Nakagawa and Matsumoto (2002) also search for exceptional elements, using support vector machines (SVMs), with greater success. Neither method has been adapted for syntactic annotation; if they are modified to do so, one has to consider what features are relevant, for which the present work provides some insight.

The idea of using odd POS sequences as indicators of problematic data is not new. Květon and Oliva (2002) employ the notion of an invalid bigram to locate corpus positions with annotation errors. An invalid bigram is a POS-tag sequence that cannot occur in a corpus, and the set of invalid bigrams is derived from the set of possible bigrams occurring in a hand-cleaned sub-corpus, as well as from linguistic intuition. Using this method, Květon and Oliva (2002) report finding 2661 errors in the NEGRA corpus (containing 396,309 tokens). Our use of rare bigrams as flags of ad hoc rules can be seen as an extension of this method. We do not limit ourselves to POS bigrams, however, using whatever sequences of nonterminal/preterminal categories are on the RHS of a rule, and no prior hand-cleaning is required.

An orthogonal line of research for detecting erroneous syntactic constructions stems from the variation n-gram method for finding inconsistencies in annotation (Dickinson and Meurers, 2003, 2005a, Boyd et al., 2008). The approach is based on a method for detecting strings which occur multiple times in the corpus with varying annotation, the *variation nuclei*. Every variation detected in the annotation of a nucleus is classified as either an annotation error or as a genuine ambiguity. The basic heuristic for detecting annotation errors requires one word of recurring context on each side of the nucleus. The nucleus with its repeated surrounding context is referred to as a *variation n-gram*. While this is useful for detecting inconsistencies in different kinds of corpora,

it misses certain types of more abstract errors, as it is string-based (see Loftsson, 2009). Thus, it seems to be more difficult to apply the method to detecting parser errors, whereas the approach explored here has a much greater chance, as demonstrated by Dickinson (2010). Similarly, Loftsson (2009) uses pattern matching based on a shallow parser to detect errors, with good success. Instead of writing patterns, however, our method automatically determines which syntactic units are erroneous.

The work presented in this paper also has connections to feature representations in the parsing literature. Comparing entire valency structures approaches rule representations in an opposite direction as is done with horizontal Markovization (Klein and Manning, 2003), though both deal with rule generalizations. For horizontal Markovization, rules are broken down into their component parts, conditioning each element on some history within the same rule. This method allows a parser to posit rules not originally observed within the training data. Using entire valency structures is a complementary way of viewing the data. Indeed, parse reranking models often employ more elaborate feature sets than those employed in standard parsing (see, e.g., Charniak and Johnson, 2005). Collins and Koo (2005), for example, use bigrams of categories as features. The current work shows that, while these properties (e.g., bigrams) can be used as model features, they have the potential to be used more directly to detect problematic parses.

Additionally, extracting lexicalized grammars from a coarse treebank—such as efforts to extract head-driven phrase grammar (HPSG) or combinatory categorial grammar (CCG) grammars (e.g., Miyao et al., 2004, O'Donovan et al., 2005, Hockenmaier and Steedman, 2007)—involves knowing whether the valency structures are correct. A potential benefit of our method is that it can assist in the preprocessing of treebanks, for example, detecting structures which lead to a failure of "grammar acquisition" (Miyao et al., 2004). Sorting noisy from rare rules (or lexical types) is important, as the number of lexical types continues to grow when adding more treebank data (see figures 4 and 5 and surrounding discussion in Hockenmaier and Steedman, 2007). Future work can also explore determining the extent to which our equivalence classes correspond to basic structures in a more well-articulated formalism.

8 Conclusions

We have presented work on detecting ad hoc rules in treebanks, where an ad hoc rule is an annotation error, covers an ungrammatical sentence, reveals issues with the uniformity of an annotation scheme or is simply a rule that does not generalize well. We started with a no-

tion of equivalence classes, the idea that different rules express the same linguistic content, with respect to valency, and then moved on to more general similarity metrics. We used two main methods, one relying on a comparison to complete representations of all other rules in a grammar (the whole daughters method) and the other relying on more local representations (the bigram method). The techniques are applicable across different treebanks, including those containing discontinuous constituents, and were shown to be effective in predicting which rules are less likely to be used in new data.

The methods are independent of treebank, language, or annotation scheme, and thus they can immediately be applied to new treebanks, including dependency treebanks (Dickinson, 2010). Indeed, given that ad hoc rule detection turns up a range of cases which provoke annotation scheme discussions, the methods are ideal for treebank development. Refining the notion of a grammar for annotation scheme-specific constructions may even lead to more precise exploration of treebank annotation.

One important area to consider in the future is that of porting grammars from one domain to another (e.g., Foster and van Genabith, 2008, Dickinson and Foster, 2009). The notion of generalizability predicts that some rules will be more applicable across domains, and with our methods, one has the potential to better determine which parts of a grammar are more domain-specific.

Our methods are also applicable to detecting errors in automatically-parsed data, by finding parsed rules which do not fit with the training grammar (Dickinson, 2010). This can complement other work which provides linguistically-interpretable error information about parsed structures, such as the approach in Goldberg and Elhadad (2010), which identifies structural biases of parsers to inspect trends in parser errors.

Acknowledgments

I would like to thank Detmar Meurers, Adriane Boyd, Jennifer Foster, and Sandra Kübler for discussion at various points throughout this work, as well as the anonymous reviewers and the many individuals who gave extremely helpful feedback at the venues where parts of this work were first presented. Portions of this work were supported by the National Science Foundation under Grant No. IIS-0623837.

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.

Bond, Francis, Sanae Fujita, Chikara Hashimoto, Kaname Kasahara, Shigeko Nariyama, Eric Nichols, Akira Ohtani, Takaaki Tanaka, and Shigeaki Amano. 2004. The hinoki treebank: Toward text understanding. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora (LINC-04)*, pages 7–10. Geneva.

Boyd, Adriane. 2007. Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of the Linguistic Annotation Workshop (LAW 2007)*, pages 41–44. Prague, Czech Republic.

Boyd, Adriane, Markus Dickinson, and Detmar Meurers. 2008. On detecting errors in dependency treebanks. *Research on Language and Computation* 6(2):113–137.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT-02)*, pages 24–41. Sozopol, Bulgaria.

Bunt, Harry. 1996. Formal tools for describing and processing discontinuous constituency structure. In H. Bunt and A. van Horck, eds., *Discontinuous Constituency*, pages 63–83. Berlin and New York: Mouton de Gruyter.

Charniak, Eugene. 1996. Tree-bank grammars. Tech. Rep. CS-96-02, Department of Computer Science, Brown University, Providence, RI.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 173–180. Ann Arbor, MI, USA.

Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1):25–69.

Daelemans, Walter, Antal van den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning* 34:11–41.

Daniels, Mike and W. Detmar Meurers. 2004. Gidlp: A grammar format for linearization-based hpsg. In *Proceedings of the Eleventh International Conference on HPSG*, pages 93–111. Stanford: CSLI Publications.

Déjean, Hervé. 2000. How to evaluate and compare tagsets? a proposal. In *Proceedings of the 2nd International Language Resources and Evaluation Conference (LREC-00)*. Athens.

Dickinson, Markus. 2005. *Error detection and correction in annotated corpora*. Ph.D. thesis, The Ohio State University.

Dickinson, Markus. 2006. Rule equivalence for error detection. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT 2006)*, pages 187–198. Prague, Czech Republic.

Dickinson, Markus. 2008. Ad hoc treebank structures. In *Proceedings of ACL-08: HLT*, pages 362–370. Columbus, Ohio.

Dickinson, Markus. 2009. Similarity and dissimilarity in treebank grammars. In *Current Issues in Unity and Diversity of Languages: Collection of the papers selected from the 18th International Congress of Linguists (CIL18)*, pages 1597–1611. Seoul.

Dickinson, Markus. 2010. Detecting errors in automatically-parsed dependency relations. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*. Uppsala, Sweden.

Dickinson, Markus and Jennifer Foster. 2009. Similarity rules! exploring methods for ad-hoc rule detection. In *Proceedings of the Seventh Workshop on Treebanks and Linguistic Theories (TLT-7)*, pages 147–158. Groningen, The Netherlands.

Dickinson, Markus and Chong Min Lee. 2009. Modifying corpus annotation to support the analysis of learner language. *CALICO Journal* 26(3):545–561.

Dickinson, Markus and W. Detmar Meurers. 2003. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT-03)*, pages 45–56. Växjö, Sweden.

Dickinson, Markus and W. Detmar Meurers. 2005a. Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 322–329. Ann Arbor, MI, USA.

Dickinson, Markus and W. Detmar Meurers. 2005b. Prune diseased branches to get healthy trees! how to find erroneous local trees in a treebank and why it matters. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT-05)*, pages 41–52. Barcelona, Spain.

Dubey, Amit. 2004. *Statistical Parsing for German: Modeling syntactic properties and annotation differences*. Ph.D. thesis, Saarland University, Germany.

Elworthy, David. 1995. Tagset design and inflected languages. In *Proceedings of the ACL-SIGDAT Workshop*. Dublin.

Eskin, Eleazar. 2000. Automatic corpus correction with anomaly detection. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*, pages 148–153. Seattle, Washington.

Foster, Jennifer. 2007. Real bad grammar: Realistic grammatical description with grammaticality. *Corpus Linguistics and Linguistic Theory: Special Issue on Grammar without Grammaticality* 3(1):73–86.

Foster, Jennifer. 2010. "cba to check the spelling": Investigating parser performance on discussion forum posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384. Los Angeles.

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 the Workshop on ROMAND 2006:Robust Methods in Analysis of Natural language Data*, pages 25–32.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Cambridge, MA: Harvard University Press.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*. Pittsburgh, PA.

Goldberg, Yoav and Michael Elhadad. 2010. Inspecting the structural biases of dependency parsing algorithms. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*.

Habash, Nizar, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitch Marcus. 2007. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1084–1092. Prague, Czech Republic.

Hall, Keith and Václav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology (IWPT-05)*, pages 42–52. Vancouver, British Columbia.

Han, Chung-hye, Na-Rare Han, Eon-Suk Ko, and Martha Palmer. 2002. Development and evaluation of a korean treebank and its application to nlp. In *Proceedings of the 3rd International Language Resources and Evaluation Conference (LREC-02)*. Las Palmas, Canary Islands, Spain.

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

Hinrichs, Erhard, Julia Bartels, Yasuhiro Kawata, Valia Kordoni, and Heike Telljohann. 2000. The tübingen treebanks for spoken german, english, and japanese. In W. Wahlster, ed., *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 552–576. Berlin: Springer.

Hockenmaier, Julia. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, The University of Edinburgh.

Hockenmaier, Julia. 2006. Creating a ccgbank and a wide-coverage ccg lexicon for german. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 505–512. Sydney, Australia.

Hockenmaier, Julia and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Hogan, Deirdre. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, pages 680–687. Prague, Czech Republic.

Hollingshead, Kristy, Seeger Fisher, and Brian Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP-05)*, pages 787–794. Vancouver, British Columbia, Canada.

Jackendoff, Ray. 1977. *X' Syntax: A Study of Phrase Structure*. Cambridge, MA: MIT Press.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 423–430. Sapporo, Japan.

Krotov, Alexander, Mark Hepple, Robert Gaizauskas, and Yorick Wilks. 1998. Compacting the penn treebank grammar. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1 (COLING-ACL-98)*, pages 699–703. Montreal, Quebec, Canada.

Kübler, Sandra. 2005. How do treebank annotation schemes influence parsing results? or how not to compare apples and oranges. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-05)*. Borovets, Bulgaria.

Kübler, Sandra, Erhard W. Hinrichs, and Wolfgang Maier. 2006. Is it really that difficult to parse german? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-06)*, pages 111–119. Sydney, Australia.

Květon, Pavel and Karel Oliva. 2002. Achieving an almost correct pos-tagged corpus. In *Text, Speech and Dialogue (TSD)*, pages 19–26.

Loftsson, Hrafn. 2009. Correcting a POS-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 523–531. Athens, Greece.

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 the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 337–344. Sydney, Australia.

Metcalf, Vanessa and Adriane Boyd. 2006. Head-lexicalized pcfgs for verb subcategorization error diagnosis in icall. Talk given at Workshop on Interfaces of Intelligent Computer-Assisted Language Learning (IICALL); The Ohio State University; Columbus, OH.

Meurers, Walt Detmar. 2005. On the use of electronic corpora for theoretical linguistics. case studies from the syntax of german. *Lingua* 115(1):1619–1639.

Miyao, Yusuke, 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 IJCNLP 2004*, pages 684–693. Hainan Island, China.

Nakagawa, Tetsuji and Yuji Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 17th International Conference on Computational Lingusitics (COLING 2002)*, pages 709–715. Taipei, Taiwan.

O'Donovan, Ruth, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the penn-ii and penn-iii treebanks. *Computational Linguistics* 31(3):329–365.

Oepen, Stephan, Dan Flickinger, and Francis Bond. 2004. Towards holistic grammar engineering and testing—grafting treebank maintenance into the grammar revision cycle. In *Beyond Shallow Analyses—Formalisms and Statistical Modelling for Deep Analysis (Workshop at The First International Joint Conference on Natural Language Processing (IJCNLP-04))*. Hainan, China.

Padro, Lluis and Lluis Marquez. 1998. On the evaluation and comparison of taggers: the effect of noise in testing corpora. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2 (COLING-ACL-98)*, pages 997–1002. Montreal, Quebec, Canada.

Petrov, Slav, 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 (COLING-ACL-06)*, pages 433–440. Sydney, Australia.

Plaehn, Oliver. 2004. Computing the most probable parse for a discontinuous phrase structure grammar. In H. Bunt, J. Carroll, and G. Satta, eds., *New Technologies in Parsing Technology*, pages 91–106. Kluwer Academic Publishers.

Przepiórkowski, Adam. 2006. What to acquire from corpora in automatic valence acquisition. In V. Koseska-Toszewa and R. Roszko, eds., *Semantyka a konfrontacja jezykowa, tom 3*, pages 25–41. Warszawa: Slawistyczny Ośrodek Wydawniczy PAN.

Rambow, Owen. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340. Los Angeles, California: Association for Computational Linguistics.

Rimell, Laura and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 475–484. Honolulu, Hawaii.

Rosén, Victoria, Koenraad de Smedt, Helge Dyvik, and Paul Meurer. 2005. Trepil: Developing methods and tools for multilevel treebank construction. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT 2005)*, pages 161–172. Barcelona, Spain.

Santorini, Beatrice. 1990. Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing). Tech. Rep. MS-CIS-90-47, The University of Pennsylvania, Philadelphia, PA.

Sekine, Satoshi. 1997. The domain dependence of parsing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, pages 96–102. Washington, DC, USA.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, pages 88–95. Washington, DC, USA.

Ule, Tylman and Kiril Simov. 2004. Unexpected productions may well be errors. In *Proceedings of the 4th International Language Resources and Evaluation Conference (LREC-04)*, pages 1795–1798. Lisbon, Portugal.

Vadas, David and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, pages 240–247. Prague, Czech Republic.

van Noord, Gertjan and Gosse Bouma. 2009. Parsed corpora for linguistics. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 33–39. Athens.

Volk, Martin. 1996. Parsing with id/lp and ps rules. In *Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference (Bielefeld)*, pages 342–353. Berlin: Mouton de Gruyter.

Voutilainen, Atro and Timo Järvinen. 1995. Specifying a shallow grammatical representation for parsing p urposes. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics (EACL-95)*, pages 210–214. Dublin, Ireland.

Wagner, Joachim, Jennifer Foster, and Josef van Genabith. 2007. A comparative evaluation of deep and shallow approaches to the automatic detection

of common grammatical errors. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 112–121. Prague, Czech Republic.

Zhao, Yanyan, Bing Qin, Shen Hu, and Ting Liu. 2010. Generalizing syntactic structures for product attribute candidate extraction. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-10)*, pages 377–380. Los Angeles, California.