On Detecting Errors in Dependency Treebanks

Adriane Boyd
The Ohio State University
adriane@ling.osu.edu

Markus Dickinson
Indiana University
md7@indiana.edu

Detmar Meurers
Universität Tübingen
dm@sfs.uni-tuebingen.de

Abstract

Dependency relations between words are increasingly recognized as an important level of linguistic representation that is close to the data and at the same time to the semantic functor-argument structure as a target of syntactic analysis and processing. Correspondingly, dependency structures play an important role in parser evaluation and for the training and evaluation of tools based on dependency treebanks. Gold standard dependency treebanks have been created for some languages, most notably Czech, and annotation efforts for other languages are under way. At the same time, general techniques for detecting errors in dependency annotation have not yet been developed.

We address this gap by exploring how a technique proposed for detecting errors in constituency-based syntactic annotation can be adapted to systematically detect errors in dependency annotation. Building on an analysis of key properties and differences between constituency and dependency annotation, we discuss results for dependency treebanks for Swedish, Czech, and German. Complementing the focus on detecting errors in dependency treebanks to improve these gold standard resources, the discussion of dependency error detection for different languages and annotation schemes also raises questions of standardization for some aspects of dependency annotation, in particular regarding the locality of annotation, the assumption of a single head for each dependency relation, and phenomena such as coordination.

1 Introduction

There is increasing interest in dependency annotation for natural language processing, which can tap into the rich history of dependency-based linguistic analysis for Slavic and other languages. Syntactic comparison measures based on constituency bracketing have been recognized as problematic for the evaluation of parsers, and

the focus has shifted to a key result of syntactic analysis, the functor-argument structure, which dependencies capture (cf., e.g., Carroll et al. 2003, Lin 2003). At the same time, direct dependency parsing approaches have been developed (cf., e.g., Eisner 1996, Tapanainen and Järvinen 1997, Nivre 2006, McDonald and Pereira 2006, Bick 2006). For both types of approaches, the dependency-based evaluation of parsers and the training and evaluation of dependency parsers, the availability of corpora with gold standard dependency annotation is crucial.

The annotation of gold standard corpora is generally the result of a manual or semi-automatic mark-up process. The annotation thus can contain annotation errors from automatic (pre-)processes, human post-editing, or human annotation. The presence of errors in linguistic annotation has been shown to create problems for both computational and theoretical linguistic uses of such corpora, from unreliable training and evaluation of natural language processing technology (cf., e.g., Padro and Marquez 1998, van Halteren 2000, van Halteren et al. 2001, Dickinson and Meurers 2005b, Habash et al. 2007, Hogan 2007) to low precision and recall of queries for already rare linguistic phenomena (cf., e.g., Meurers and Müller 2008). Investigating the quality of linguistic annotation and improving the annotation and the annotation schemes where possible is thus a key issue for the use of annotated corpora in computational and theoretical linguistics. Even a small numbers of errors can have a significant impact on the uses of linguistic annotation. For example, Habash et al. (2007) compare Arabic parsing systems and find that the one which initially had better results was actually worse when treebank errors were accounted for. In another case, Hogan (2007) identifies specific types of coordinate phrases which are inconsistently annotated. She points out that they cause problems for training and testing and cleans up these cases to improve performance. When one considers that linguistic annotation such as part-of-speech or syntactic annotation is often used as input for further layers of annotation or processing, even a small set of errors can significantly affect a higher layer. For example, the errors in syntactic rules identified in Dickinson and Meurers (2005b) are often attributable to erroneous part-of-speech annotation.

The idea that variation in annotation can indicate annotation errors has been explored to detect errors in part-of-speech annotation (van Halteren 2000, Dickinson and Meurers 2003a) and, to a lesser extent, in syntactic annotation (Ule and Simov 2004, Dickinson and Meurers 2003b, 2005a,b). But, as far as we are aware, the research we report on here is the first general approach to error detection for dependency treebanks, whose annotations are typically less local and also differ in other interesting respects from constituency-based treebanks.[1] While an error in

[1] Inter-annotator agreement testing and annotation specific validity checks are, of course, used during the construction of such resources (cf., e.g., King et al. 2003, sec. 2.1), but we are not aware of automated, general error detection approaches.

constituency annotation might reflect a bracketing decision without major consequences for the interpretation of the sentence, incorrect dependency relations will generally affect the interpretation of the sentence. On the other hand, dependency annotation does not need to commit to certain attachment ambiguities plaguing constituency annotation.

In this context it is also relevant that the current work on dependency annotation builds on a range of long-standing linguistic traditions of analysis (e.g., Tesnière 1959, Mel'čuk 1988, Sgall et al. 1986, Hudson 1990), not a single accepted standard. While there are some fundamental assumptions that all dependency grammar approaches share – such as the primacy of word-word relations and functional structure – they can vary widely in other assumptions, and thus also in the resulting annotations (cf, e.g., the discussion in Nivre 2005). Such divergence particularly arises in the analysis of phenomena which run counter to the reliance on word-word relations, such as in the analysis of coordination or adverbial scope, where sentential adverbs need to be distinguished from those modifying only the verb. For coordination, which we turn to in section 3.2, some analyses even go as far as positing a limited amount of syntactic constituency (Hudson 1990). In a related way, the tenets of some dependency approaches, such as the single-head constraint also discussed in section 3.2 (cf, also Mel'čuk 1988, ch. 1), are relaxed in other approaches, e.g., to explicitly encode all dependencies in control constructions (as, e.g., in the TigerDB of Forst et al. 2004).

For the research on detecting errors in dependency annotation presented in this paper, the fact that dependency analyses vary across different annotated corpora in significant ways means that care must be taken to distinguish those aspects which relate to dependency annotation in general from those where more specific assumptions are involved. In detecting problematic annotations in divergent corpora, the error detection method to be presented also provides feedback on the further development of dependency annotation schemes. In addition to its main focus on developing a practical method for detecting errors in gold standard dependency resources, this paper thus also contributes to advancing the understanding of dependency annotation, its standardization, and its relation to constituency-based annotation.

2 Background: Variation detection

As starting point for developing an error detection method for dependency annotation, we use the variation *n*-gram approach for constituency-based treebanks developed in Dickinson and Meurers (2003b, 2005a). The approach is based on an efficient method for detecting strings which occur multiple times in the corpus with varying annotation, the so-called *variation nuclei*. For example, Figure 1 shows the

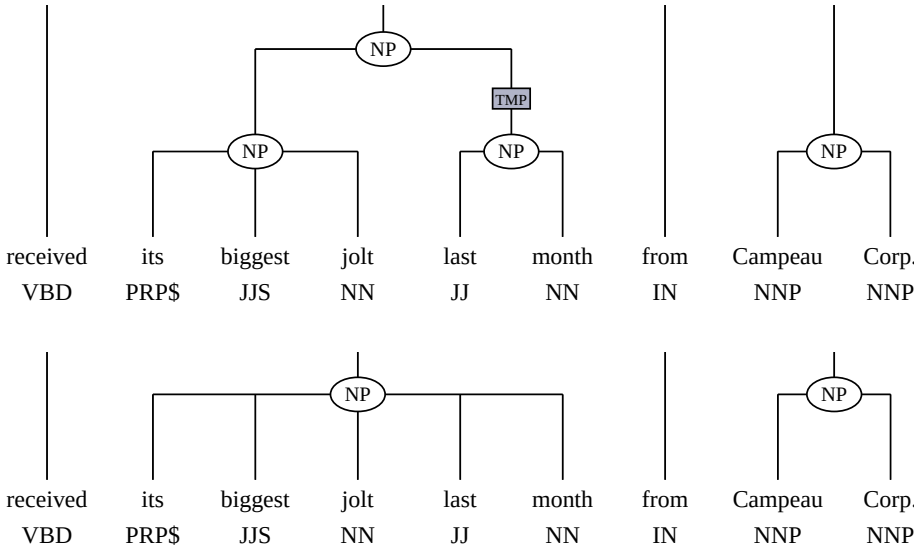


Figure 1: Bracketing error detected in the Penn Treebank

variation nucleus *last month*, which occurs twice in the Wall Street Journal portion annotated in the Penn Treebank (Taylor et al. 2003).

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 the original proposal in Dickinson and Meurers (2003a) is to expand the context as far as possible given the repeated *n*-gram, using only the immediately surrounding words as context is sufficient for detecting errors with high precision (Dickinson 2005, pp. 46, 81). This local context heuristic receives independent support from current research on first language acquisition showing that such context frames are used by infants during lexical category learning (Mintz 2003, 2006). Similarly, work on unsupervised grammar induction (cf. Klein and Manning 2002) also emphasizes the importance of local syntactic context.

The approach can detect bracketing and labeling errors in constituency annotation. The variation nucleus *last month* we saw in Figure 1 occurs once with the label NP and once as a non-constituent, which in the algorithm is handled through a special label NIL. As an example of a labeling error, the variation nucleus *next Tuesday* occurs three times in the Penn Treebank, twice labeled as NP and once as PP (Dickinson and Meurers 2003b).

The basic method for detecting annotation errors in constituency-based syntactic annotation was extended to discontinuous constituency annotation in Dickinson

and Meurers (2005a). Supporting discontinuity brings it in line with one of the important characteristics of dependency annotation. In the next section, we turn to a discussion of this and other key aspects of dependency annotation that an error detection approach for dependency annotation needs to take into account.

3 Key aspects of dependency annotation

We discuss the error detection approach for dependency annotation on the basis of three diverse dependency annotation schemes that have been used to annotate corpora for three different languages: the Talbanken05 corpus of Swedish (Nivre et al. 2006), the Prague Dependency Treebank (PDT 2.0) of Czech (Hajič et al. 2003), and the Tiger Dependency Bank (TigerDB) of German (Forst et al. 2004).

The Swedish treebank Talbanken05 is a reconstruction of the Talbanken76 corpus (Einarsson 1976a,b), which consists of text drawn from written data (professional prose, student essays) and spoken data (interviews, conversations, debates). After deepening the original grammatical function and flat phrase structure annotation, it was converted to dependency structures (Nilsson and Hall 2005). Talbanken05 consists of a total of 21,571 sentences with approximately 320,000 tokens, which are annotated using 69 types of dependency relations. The original corpus was manually annotated and has undergone a conversion process that is "very reliable" (Nivre et al. 2006), resulting in a high-quality corpus.

The Prague Dependency Treebank (PDT 2.0, http://ufal.mff.cuni.cz/pdt2.0/) is composed of text from newspapers, business weeklies, and scientific journals and includes annotation on morphological, analytical, and tectogrammatical layers. We focus on the analytical layer, a dependency graph representation of the surface syntactic structure which distinguishes 28 types of dependency relations. It consists of 1.5 million tokens in 88,000 sentences. The annotation is the result of manual annotation dating back to 1996, as well as several annotation-specific error checks (Hajič et al. 2001, secs. 3.2, 3.3), and it is generally regarded as being of very high quality.

The German Tiger Dependency Bank (TigerDB) was semi-automatically derived from the Tiger Treebank (Brants et al. 2002), a corpus of German newspaper text taken from the Frankfurter Rundschau. The subsection of the Tiger Treebank annotated in TigerDB includes 36,326 tokens in 1,868 sentences. The dependency annotation in the TigerDB is based on the annotation scheme developed for the English PARC 700 Dependency Bank (King et al. 2003). The annotation scheme distinguishes 53 types of dependency relations, which are established between words but also between sublexical and abstract nodes – an issue that we return to in section 3.2.

3.1 Overlap and contiguity in dependency vs. constituency treebanks

Let us start the discussion of the key properties of dependency annotation for our work on error detection with a basic example. Example (1) from the Talbanken05 corpus illustrates the ordered relations between words that are encoded in a dependency treebank.

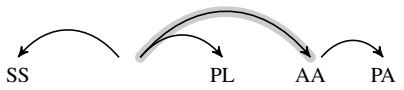
(1) Deras utbildning tar 345 dagar
Their education takes 345 days

Each arc represents a relation from a head to a dependent, with the arrow pointing to the dependent. The labels on the arcs distinguish the different types of dependencies; here the label SS expresses that *utbildning* ('education') is the subject of *tar* ('takes') and OO that *dagar* ('days') is its object; DT is a determiner dependency.

In comparing the properties of dependency annotation to the constituency treebanks for which the variation detection approach was developed, there are two issues which are important to spell out here and to which we will return in section 4. Firstly, there is what we will refer to as *overlap*. Two phrases in a constituency-based representation never share any of their daughters unless one is properly included in the other. In a dependency representation, on the other hand, the same head may well participate in multiple dependency relations, causing dependency pairs to overlap, i.e., to share exactly one token. In example (1), for example, *utbildning tar* ('education takes') and *tar dagar* ('takes days') are two distinct dependency pairs, which both contain the token *tar* ('takes'). In a constituency tree, *tar* would be part of the verb phrase *tar 345 dagar* and any other constituent including *tar* either properly contains this phrase or is contained by it. In addition to dependencies overlapping in the head token, in some dependency representations, such as the TigerDB, overlap can also arise when a given token is a dependent of multiple heads. We take a closer look at this issue in the discussion of the single-head constraint in the next section.

Secondly, there is the issue of *contiguity*. Within traditional constituency frameworks, the sisters in a local tree are contiguous, i.e., their terminal yield is a continuous string. For dependency annotation, on the other hand, a dependency graph will often relate non-contiguous elements. In sentence (2), for example, the word *ger* ('gives') is modified by *med* ('with'), with a particle appearing in between.[2]

[2]Dependency labels used: PL = verb. part., PA = compl. of prep., AA = (other) adverbial



(2) Handeln ger tillbaka med dem
Commerce gives back with them

While non-contiguity is common, a less frequent special case of non-contiguity is non-projectivity. In non-projective annotation, dependency relation arcs cross. The degree to which non-projectivity arises depends on the language and the underlying analysis; a comparison is provided in Havelka (2007). Non-projectivity requires special attention in dependency parsing (e.g., Nivre 2006, McDonald and Pereira 2006), but we will see that for our error detection method the ability to deal with non-projective annotations results from dealing with non-contiguous relations in general.

3.2 Differences in annotation schemes and their impact

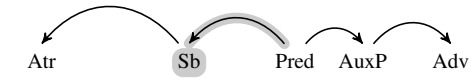
While the different approaches to dependency annotation share the properties mentioned in the previous section, they vary significantly in others (cf, e.g., Nivre 2005). Some of these differences can impact the variation *n*-gram approach to error detection – despite the fact that, as a data-driven method testing the annotation consistency for comparable strings recurring in the corpus, it is largely independent of the particular annotation scheme and language. Generally speaking, the method depends on the assumption that the annotation labels are determined by the properties of the data unit being compared (i.e., the variation nucleus) and that some relatively local context is sufficient to distinguish genuine ambiguities from variations which are annotation errors – two assumptions to which there are exceptions in some dependency annotation schemes.[3]

Dependency relations encoded indirectly As just characterized, the variation *n*-gram method relies on the assumption that the label of the variation nucleus can be determined based on the recurring string, the nucleus surrounded by a limited number of context words. One aspect of this strictly data-driven perspective is that each dependency relation label is considered independent of the others—akin to the well-known independence assumption for local trees in PCFGs (cf, e.g., Manning and Schütze 1999, ch. 11). This is not the case, however, for certain aspects of dependency treebanks where annotation decisions are based upon annotation decisions elsewhere in the graph. We will use the term *indirect* to refer to those

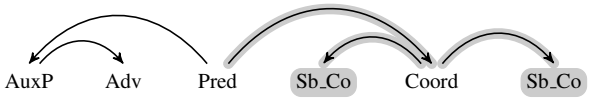
[3]Related issues, of course, arise in other linguistic frameworks as well; given the focus of the paper we here discuss them for dependency annotation only.

annotations which are not determined by the two words being annotated but by something beyond the direct dependency.

As an example for where indirect annotation affects the comparability of the annotation for the same recurring string, consider the case of coordination. Coordination is often given a special status in dependency representations, due to the challenges it presents. In the PDT, for example, conjuncts are dependent upon the conjunction and a special suffix is appended to their label. For example, the predicate *jsou* ('are') in (3a) selects its subject *telefony* ('telephones'). But the same verb in (3b) does not establish a direct dependency relation with its subject. Instead, it selects a coordination, which then selects the subject dependent of the verb, using the Sb label to which the Co suffix has been added. In other words, the Sb part of the label indirectly encodes the subject relation for the verb and the Co part of the label encodes the direct dependency on the coordination. As a consequence of this setup, the subject *telefony* ('telephones') also receives two different labels in the two examples, Sb in (3a) and Sb_Co in (3b).



(3) a. Nejlevnější telefony jsou v Británii
cheapest telephones are in Britain



b. Na pokojích jsou telefony a faxy
in rooms are telephones and fax machines

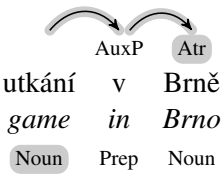
The analysis for the labels of coordinated elements, using a Co suffix together with an indirectly determined function, such as the Sb in the above example, is one of several phenomena in which the dependency labels are not determined locally in the PDT.

Another common example for such a phenomenon is the selection of prepositional phrase dependents, where the preposition always bears the special label AuxP and the dependency label specifying the function of the prepositional phrase is shown on the noun selected by the preposition.[4] This is exemplified in (4) where in (4a) the noun *utkání* ('game') occurs with an attribute (Atr) dependent, but only establishes this dependency indirectly through the preposition *v* ('in') as a depen-

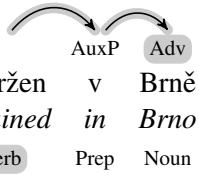
[4]This indirect encoding is limited to the analytical layer of the PDT annotation. The more theoretically-motivated tectogrammatical layer of the PDT annotates the relation between the head of a preposition and its argument directly (Mikulová et al. 2006).

dent with the special label AuxP.[5] The corresponding case with an adverbial (Adv) dependent that is indirectly selected is shown in (4a).[6]

(4) a.

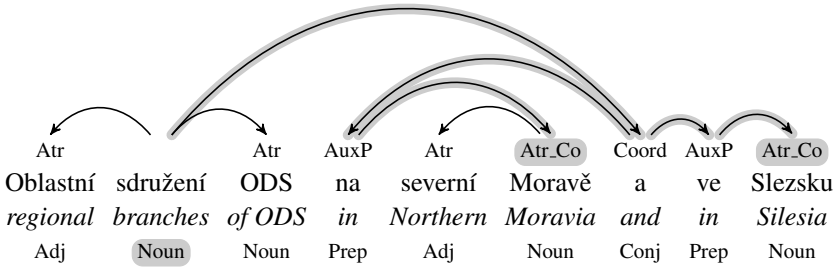


b.

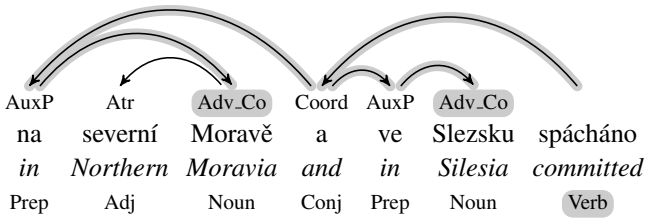


Relevant in our context of determining which local contexts are sufficient for identifying a label variation as an annotation error, several of these phenomena can occur together in a single sentence so that a head can be several dependencies removed from the dependency label it determines. Take, for example, the noun *sdružení* ('branches') in (5a). It in principle occurs with an attribute (Atr) dependent, but in this sentence its direct dependent is the conjunction bearing the label Coord, which then selects two prepositions with dependency label AuxP, which then select the nouns *Moravia* and *Slezsku*, which bear the Atr label together with the Co suffix. The head noun *sdružení* thus is two dependencies removed from the Atr label it selects.

(5) a.



b.



[5]Here and in other PDT examples where it is relevant, the bottom line shows the part of speech encoded in position 1 of the morphological tag (cf. Hana and Zeman 2005, p. 15).

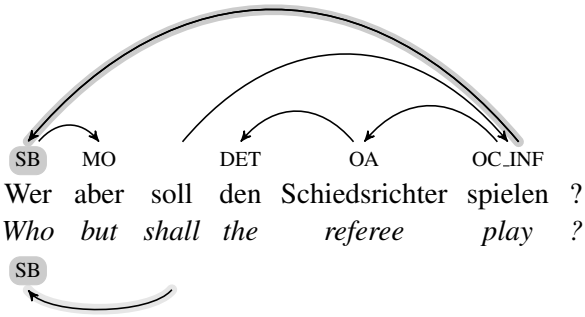
[6]An alternative to establishing such dependencies indirectly would be to analyze the noun as a dependent of both the preposition and the noun/verb taking the prepositional phrase dependent. There thus is a tradeoff between using indirectly encoded dependencies and relaxing the single head constraint as two different options for encoding the dependencies for such phenomena.

In (5b), we have the corresponding case with the verb *spácháno* ('committed'), which in principle occurs with an adverbial (Adv) dependent, but in this example only establishes this relation indirectly given the intervening coordination and preposition nodes. Even though in both examples in (5), *Moravě* and *Slezsku* appear in the identical recurring string *na severní Moravě a ve Slezsku*, their dependency label differs because of the indirect annotation scheme decisions.

It is important to keep in mind here that the issue discussed is not an idiosyncrasy of one specific dependency bank. The annotation of coordination and prepositional phrases intrinsically is a hard analytic issue for a dependency analysis (cf., e.g., the discussion in Nivre 2005, p. 11).[7]

While in general dependencies which are encoded indirectly can lead to low precision for an error detection method which is based on identifying variation in the annotation of locally identical strings, in the discussion of the results for the PDT in section 5 we show that in practice this issue arises relatively infrequently and that a local switch of dependency labels is sufficient to eliminate those false positives.

Single-head constraint Returning to the issue of overlap, dependency annotation schemes also differ in whether they assume a single head or not. Typically, each dependent is required to depend on a single head. However, some dependency annotation schemes allow for a word to be dependent upon multiple heads, creating more overlapping structures. Example (6) from the TigerDB, translating as *'But who shall function as the referee?'*, illustrates this situation.



(6) Wer aber soll den Schiedsrichter spielen ?
Who but shall the referee play ?

In this sentence, the subject *wer* ('who') is analyzed as being dependent upon both the modal verb *soll* ('shall'), for which it is the syntactic subject, and the main verb

[7]Related issues can arise in a constituency analysis, as exemplified by the use of the CNP label for coordinated noun phrases in the Tiger Treebank (Brants et al. 2002) or the use of topological field nodes intervening between a head and its dependent in TüBa-D/Z (Telljohann et al. 2005). However, indirect grammatical function assignment can be avoided in more cases given that a constituency analysis can postulate additional constituency nodes to group the material which together realizes a grammatical function.

spielen ('play'), for which it is a semantic argument.

In order to be able to work with a range of dependency treebanks, an error detection method thus cannot assume single-headedness. Since our method deals with each mapping from a nucleus to its annotation independently, the approach is independent of the number of dependency relations a word participates in and whether some of those relate it to different heads.

Level of abstraction Finally, dependency treebanks also differ in their relationship to the surface string, particularly in their treatment of word forms and linear order. A dependency analysis can involve unordered relations between lemmas, as in the TigerDB, where lemmas and abstract nodes with or without lexical counterparts are the elements involved in dependency relations, and the relationship to the surface string must be re-created. Furthermore, some words, such as auxiliary verbs functioning as tense markers, are left out of the dependency graph. Annotating dependency relations for such abstract representations instead of for the surface string is problematic for data-driven methods, such as the variation *n*-gram approach. We return to the TigerDB and how the method can be applied to such abstract representations in section 5.

4 Variation detection for dependency annotation

Having introduced the idea of variation detection for constituency-based annotation and the aspects of dependency annotation of direct relevance in this context, we are ready to connect the two by exploring what is involved in applying the variation *n*-gram method to dependency annotation. There are two questions to consider: Firstly, what constitutes the comparable recurring units of data that are annotated, i.e., what are the variation nuclei for dependency annotation? Secondly, what are appropriate and sufficient disambiguating contexts for distinguishing genuine ambiguity from erroneous variation when dealing with dependency annotation?

4.1 Determining the variation nuclei

The starting point of determining the recurring comparable units which vary in their annotation is straightforward: A dependency relates exactly two words, so the method needs to investigate the mapping between a pair of words and their dependency relation label.

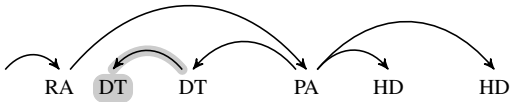
Algorithmically, the variation *n*-gram approach for discontinuous syntactic annotation error detection (Dickinson and Meurers 2005a) is the approach we are adopting, given that dependencies are not restricted to relate contiguous words. Since we are dealing with labeled dependency pairs, the algorithm only needs to

consider variation nuclei of size two. Before spelling out the algorithm, several additional aspects need to be clarified, though, to which we turn in the following.

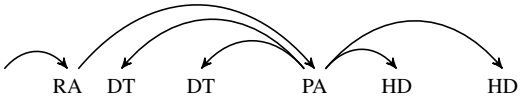
Encoding directedness of dependency relations Different from the category labels of constituents, dependency arcs are directed. To encode which of the two words is the head of the dependency, we encode the direction in the label, adding an L in case the head is to the left and an R if it is to the right. For example, for the Swedish sentence we saw in (1) on p. 6, we have the nuclei *Deras utbildning* labeled DT-R, *utbildning tar* (SS-R), *tar dagar* (OO-L), and *345 dagar* (DT-R).

With the dependency-annotated data encoded in this way, variation *n*-gram detection can distinguish variation in dependency labeling (e.g., SUBJ-R vs. OBJ-R), and variation in the determination of the head (e.g., OBJ-L vs. OBJ-R).

NIL occurrences Adapting the use of the special NIL label for bracketing mismatches in constituency-based annotation discussed in section 2, we can also detect variation in dependency identification, i.e., the situation where two words which are related by a dependency in one occurrence in the corpus are compared to an occurrence of those words that is not annotated with a dependency (e.g., OBJ-R vs. NIL). For example, the string in example (7) appears twice in the Talbanken05 corpus.[8]



(7) a. Backberger säger i **sin** **artikel** ’ Den heliga familjen …
Backberger says in her article ’ The sacred family …



b. Backberger skriver i **sin** **artikel** ’ Den heliga familjen …
Backberger writes in her article ’ The sacred family …

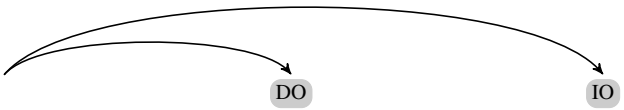
In one instance, *sin* (‘her’) is annotated as the determiner (DT) of *artikel* (‘article’); in the other, there is no relationship between *sin* and *artikel*, as both are dependents of another head. The method assigns the second occurrence the label NIL, making it possible to detect a variation in dependency identification here, between the labels DT-R and NIL.

The search for NIL occurrences of dependency pairs is restricted to words found within the same sentence, given that dependency labels only make reference

[8]Here and in the rest of the paper, the two words in the variation nucleus are shown in boldface.

to intrasentential relations. Additional restrictions could be imposed if additional locality domains (e.g., clause boundaries) are annotated. One could also consider using a notion like that of a *span* in Eisner (1996), in order to reduce the search space for NIL nuclei, since spans delimit "grammatically inert" portions of the sentence. Such a domain restriction, however, would limit us to finding only those NIL nuclei that correspond to projective dependencies, as a word in the middle of a span in the sense of Eisner (1996) cannot relate to a word outside that span.

Dealing with overlap In addition to taking into account the non-contiguity and directedness of the dependencies, further differences between constituency and dependency annotation also need to be accounted for when determining which occurrences of two words are comparable, i.e., which nuclei are worth inspecting. The overlap of dependency relations mentioned in section 3 raises the question of how to handle the annotations of type-identical overlapping pairs. For example, consider the two objects *chair* of the verbal head *showed* in (8).



(8) She showed the department chair the beautiful old chair .

The pairs are type-identical and their head is the same token, *showed*. Clearly, such type-identical overlapping relations should not be interpreted as nuclei that need to be compared to determine whether there is variation between the two annotations.

Our solution to this issue, when multiple type-identical dependencies share a token, is to consider all those dependency labels as a set. For example, for (8) our algorithm considers a single dependency triple <showed,chair,{DO, IO}>. This means that there is no variation within this example, but it goes further in being able to handle multiple comparable dependents in different sentences. For example, if sentence (8) occurs twice within the same corpus, we do not want to compare the dependency relation with the direct object in one sentence to the relation with the indirect object in the other. Thus, for overlapping type-identical dependencies, the nucleus encodes the set of dependency labels for all occurrences anchored on the same head token. Then, the set of labels are compared against another set of dependency labels to determine if they are the same. This restricts variation detection for those cases to nuclei which are meaningful to compare.

This solution will naturally not cover cases of variation across sentences, for example, between a word being used as the DO of a verbal head in one sentence and an IO of the same verbal head in another sentence. Note, though, that these cases will be filtered out by the context heuristics in section 4.2. One could also consider exploiting information about the types of common sister dependencies in

order to define valid substitution sets of sisters (e.g., that DO/IO variation is often acceptable), akin to defining valid substitution sets of mothers for constituency annotation (cf. Dickinson and Meurers 2005b). We leave this for future work, but note that it would not catch some cases that we currently identify.

Algorithm for identifying variation nuclei Having clarified the relevant issues we need to take into account when adapting the variation *n*-gram approach to dependency annotation, we are ready to spell out the algorithm for detecting the variation nuclei in dependency annotation.

1. Compute the set of nuclei:[9]
   a) Find all dependency pairs, store them with their category label.
      - The dependency relations annotated in the corpus are handled as nuclei of size two and mapped to their label plus a marker of the head (L/R).
      - The labels of overlapping type-identical nuclei are collapsed into a set of labels.
   b) For each distinct type of string stored as a dependency, search for non-dependency occurrences of that string and add the nuclei found with the special label NIL.[10]

      To obtain an algorithm efficient enough to deal with large corpora, we adopt the following measures from Dickinson and Meurers (2005a):
      - A trie data structure is used to store all potential nuclei and to guide the search for NIL nuclei.
      - The search is limited to pairs occurring within the same sentence.
      - NIL nuclei which would be type-identical to and overlap with a genuine dependency relation in the same sentence are not considered.[11]
2. Compute the set of variation nuclei by determining which of the stored nuclei have more than one label.

[9] In Dickinson and Meurers (2003b, 2005a), this step is embedded within a loop over all constituent sizes in the corpus, which is not needed here given that dependencies always relate exactly two words.

[10] In other words, we only search for (potentially discontinuous) NIL nuclei which are type-identical to genuine dependency nuclei, i.e., there is no NIL vs. NIL variation.

[11] Note that this step is distinct from the overlap of type-identical relations discussed above. The issue here is limiting the number of NIL nuclei which need to be postulated.

4.2 Determining appropriate contexts

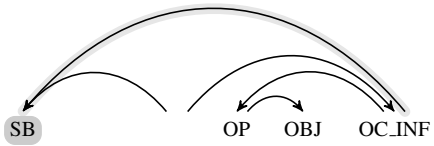
Turning from the identification of variation to determining which of those variations constitute errors and which are genuine ambiguities, we need to determine what kind of context is required to carry out this task for dependency annotation. We here discuss three heuristics, which differ in interesting ways with respect to the precision/recall tradeoff and thus become relevant in different situations (cf. section 5).

Non-fringe heuristic The variation *n*-gram method for constituency-based annotation we use as a starting point uses identical surrounding words as context: variation nuclei on the fringe of comparable strings are not reliable as indicators of errors since they lack context on one side. This basic idea is also applicable to dependency annotation. Following Dickinson and Meurers (2005a) we will refer to the heuristic requiring one element of context around each word in the nucleus as the *non-fringe heuristic*. For the two word nuclei used in dependency annotation this means that an element of a nucleus can be non-fringe by being next to a context word or next to the other word in the nucleus.

Given that this heuristic requires identical words surrounding each word in the nucleus, it detects errors with high precision.

NIL internal context heuristic A less stringent context requirement for dependency annotation can increase recall by not requiring context around every word. Dependency pairs are frequently realized non-contiguously, which gives the intervening context an added importance for disambiguation. This is most prominently the case for non-contiguous NIL pairs, i.e., the occurrence of two words in a sentence which somewhere else in the corpus occur as a labeled dependency. As explained in the discussion of example (7) in the previous section, such NIL pairs are important to take into account to detect errors in dependency identification. But without context heuristics, the number of NIL nuclei identified this way is high, containing many false positives. Identifying only the two nucleus words by themselves, in particular when they can occur non-contiguously, is insufficient for detecting errors with high precision. We therefore require NIL nuclei to have the same internal context as an annotated dependency, a restriction not imposed when comparing two nuclei with annotated dependencies. We refer to this as the *NIL internal context heuristic*.

TigerDB example (9) illustrates this with the nucleus *Wirtschaftspolitik warten*.

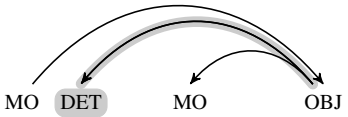


(9) a. “ **Wirtschaftspolitik** läßt auf sich **warten** ”
economic policy lets on itself wait

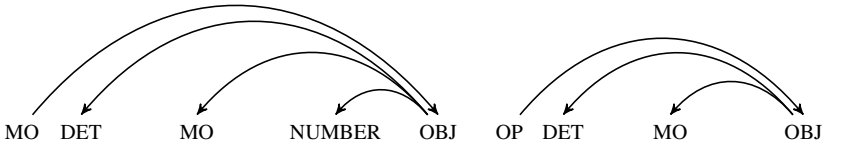
b. Die **Wirtschaftspolitik** läßt auf sich **warten** .
the economic policy lets on itself wait
‘Economic policy is a long time coming.’

In (9a), *Wirtschaftspolitik* (‘economic policy’) is the subject of *warten* (‘wait’), while in (9b) there is a NIL nucleus. The internal context between *Wirtschaftspolitik* and *warten* is identical (*läßt auf sich*), whereas the outer context differs. If one requires a context consisting of identical words around every nucleus word, our method would miss this case because the outer context is not the same. The NIL internal context heuristic allows the method to detect cases such as this one.

At the same time, the NIL internal context heuristic successfully prevents false positives which would result from relaxing the context requirements further. Consider the examples in (10), where we have a relation with *den* (‘the’) as the determiner of *Staaten* (‘States’).



(10) a. in **den** Vereinigten **Staaten**
in the United States



b. in **den** vergangenen zehn Jahren an die Vereinigten **Staaten**
in the past ten years to the United States

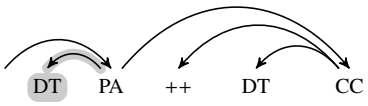
Without requiring the presence of the internal context, the error detection method

would posit a discontinuous nucleus *den Staaten* with label NIL for example (10b). The internal context filter eliminates this variation.

Dependency context heuristic Turning from the distributional context to a type of context making reference to the dependency annotation itself, consider that the head of a variation nucleus typically will be a dependent of another word in the rest of the sentence. We can use this dependency annotation connecting the variation nucleus to the rest of the sentence as a valuable, disambiguating context given that the two words in a variation nucleus are more likely to be annotated the same way if the head of the variation nucleus has the same function in the sentence in each occurrence. This is particularly relevant in dealing with the issue of indirect dependency annotation discussed in section 3.2. We thus introduce the *dependency context heuristic* which encodes the idea that if the head of a variation nucleus is being used in the same function in all instances, the variation in the labeling of the nucleus is more likely to be an error. Conversely, when the head is used differently, it is more likely a genuine ambiguity.

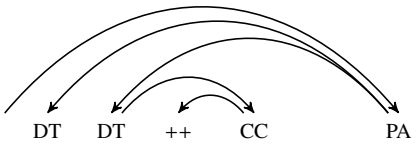
For example, consider the two examples in (11) involving the variation nucleus *den ena* being annotated as DT-R in example (11a) but not being annotated as a dependency pair, i.e., a NIL nucleus, in (11b).

(11) a.



i	**den**	**ena**	eller	andra	formen
in	*the*	*one*	*or*	*other*	*form*

b.



i	**den**	**ena**	eller	båda	färdriktningarna
in	*the*	*one*	*or*	*other*	*directions*

Under the analysis of coordination in the Talbanken05 corpus, this is acceptable variation, despite occurring in the identical string context (*i den ena eller*). Because *ena* has different functions—it is the place adverbial (PA) dependent of *i* ('in') in (11a) and a determiner (DT) dependent of *färdriktningarna* ('directions') in (11b)—the dependency context heuristic correctly categorizes it as a genuine ambiguity instead of as an annotation error.

While this heuristic increases error detection precision, it in principle can also decrease recall given that variation nuclei which differ in the selection of the head

(i.e., in the directedness of the relation) will not be detected using this heuristic. However, in practice we found that errors in directedness are very rare.[12]

5 Results and Discussion

Having defined variation nuclei and appropriate contexts for dependency annotation, we test the approach to error detection with the three corpora introduced at the beginning of section 3: the Talbanken05 corpus of Swedish, the Prague Dependency Treebank of Czech, version 2.0 (PDT2.0), and the Tiger Dependency Bank (TigerDB) of German. The underlying error detection method is the same in all cases, but we varied the heuristics used as discussed below. While detecting instances of errors as such is a primary goal, we also want to emphasize the feedback on the annotation schemes that such error detection work can provide – an aspect largely ignored in previous error detection research.

Talbanken05 We applied the error detection algorithm defined in section 4.1 to the written half of the Talbanken05 corpus (sections P and G), consisting of 197,123 tokens in 11,431 sentences. The algorithm returned 210 different variation nuclei with at least one word of context around each word in the nucleus, i.e., using the non-fringe heuristic.[13] To evaluate the precision of the error detection method, each of these cases was hand-inspected by an independent researcher familiar with the annotation scheme. In 92.9% (195) of the detected cases, a detected variation pointed to one or more errors, identifying a total of 274 token errors. For such a small corpus with a manual annotation known to be of high quality, this is a significant number of errors, and they are detected with better precision than previously found for constituency treebanks for English and German (Dickinson and Meurers 2003b, 2005a). Furthermore, the method points not only to specific errors, but to potential classes of errors that a human annotator can recognize and use to detect and correct a higher number of annotation errors – something the independent Talbanken researcher evaluating the results confirmed as indeed being the case in practice.

Turning to an analysis of the nature of the errors detected by the method, of the 274 token errors, 145 were instances of one label being incorrectly used for another. 129 were instances where there was an error in identifying the presence of a dependency, i.e., involving a NIL nucleus, with either the treebank label being

[12] For the Talbanken05 corpus we detected no such cases, in the PDT there was a single case, and in the TigerDB we found four such case.

[13] The results reported here are for a version of the algorithm which does not distinguish between heads and dependents in testing whether two n-grams cover the same tokens; additional variations can be found by adding that distinction.

incorrect or the NIL label incorrect. Errors in labeling of a dependency and errors in identifying a dependency arguably are both important to detect; they seem to occur about equally often; and the method successfully identifies both.

Examining these 274 token errors more closely, we find some consistent patterns. For one thing, determiners (DT) are far and away the most problematic category: 30 token errors are an incorrect use of DT, and 15 errors need to be corrected to DT, reflecting the fact that in Swedish a broader range of words can function as determiners. For another, 20 errors featured an erroneous complement of preposition (PA) needing to be changed to the other head (HD) category; importantly, both of these categories were introduced in the conversion from Talbanken76 to Talbanken05, and our method pinpoints these automatic conversion problems. Finally, in the verbal domain, adverbials are involved in more errors (73 cases) than arguments, i.e., subjects and objects (31 cases).

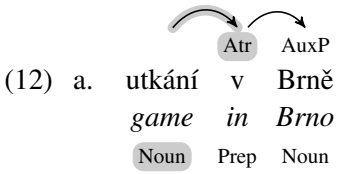
While 92.9% error detection precision clearly is high enough to drive an effective process correcting the annotation errors, it still is interesting to also determine the impact of the dependency context heuristic discussed in section 4.2. After adding this heuristic, the number of variation nuclei found by the method drops to 98, of which 95.9% (94) contain at least one error. Adding the dependency context heuristic thus increases precision, but at the cost of over 50% drop in recall.

PDT2.0 For the PDT2.0, we ran the error detection method on the subset of sentences in the *full/amw* section of the corpus, containing 38,482 sentences (670,544 tokens), almost half of the 88,000 sentences with annotation on the analytical layer.

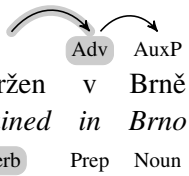
Using the non-fringe heuristic, requiring one word of context around each word in the nucleus, the algorithm detects 553 cases. To focus on cases of linguistic variation, all instances where the variation involved punctuation were removed, which leaves 426 variation nuclei.

In our discussion of the differences between annotation schemes in section 3.2, we mentioned that the PDT annotation scheme includes a number of dependency labels which are indirectly determined. A researcher familiar with the PDT annotation scheme hand-inspected a sample of the detected cases and confirmed in practice that a significant number of false positives were due to the indirectly-encoded dependencies involving AuxP and AuxC relations. To eliminate those cases before the hand evaluation of the error detection results by an independent researcher, we switched the relation between the preposition and its argument with the AuxP relation between the preposition and its head, thus directly encoding the dependence on the category of the head (noun vs. verb), as shown in (12) next to the original annotation repeated in (13).

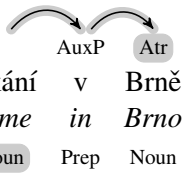
(12) a.



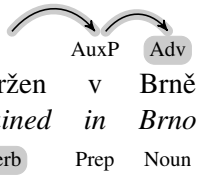
b.



(13) a.



b.



Modifying the encoding of AuxP and the parallel AuxC relations in this manner reduces the number of variations detected by the algorithm to 354 cases. The independent researcher familiar with the PDT annotation scheme hand-inspected the detected cases and determined that there were annotation errors in 59.7% (205) cases, corresponding to 251 token errors.

Inspection of the false positives revealed that 49.0% (73) are due to indirect annotation scheme decisions other than AuxP or AuxC, namely the annotation of coordination discussed in section 3.2. Modification of the annotation scheme to reduce or eliminate the use of indirectly determined dependency labels thus would significantly improve the precision of the error detection method – and of other data-driven methods making use of the annotation.

Further analysis of the other false positives provided further interesting feedback on the annotation scheme. For example, a number of the variations detected by the method are due to permitted annotator variation, such as with the dependency relations AdvAtr and AtrAdv used when a phrase could be interpreted either as an adverbial of the predicate (Adv) or as an attribute of a lower node in the tree (Atr). AtrAdv is used when the annotator has a preference for the Atr interpretation; AdvAtr is used when Adv is preferred, and the annotation guidelines state that "This decision is difficult to describe in general terms, it will depend on the annotator's attitude" (Hajič et al. 1999, p. 57). If such subjective annotation decisions are indeed a useful component of the annotation scheme, it would be straightforward to automatically eliminate such permitted variation from the set of variations found by the error detection method.

Looking at the nature of the errors detected, we found a small change from the Talbanken05 results. A higher percentage of the 251 tokens featured a labeling error, 60.6% (152 instances). In 99 instances, there was an error in identifying the presence of a dependency, nearly evenly split between the treebank label being incorrect (50) and the NIL label incorrect (49).

Finally, we again investigated the impact of another context heuristic on the er-

ror detection results. Given the 59.7% precision with the non-fringe heuristic, we were interested in whether adding the dependency context heuristic would improve the error detection precision. We found that adding this heuristic narrowed the 354 variations down to 222 cases which have the same dependency context in all instances. Of these 222, 61.7% (137) were determined to be errors. The dependency context heuristic thus is not effective for improving error detection precision in this corpus.

TigerDB The TigerDB consists of only 1,868 sentences drawn from the Tiger Treebank. As discussed in section 3.2, the annotation is based on lexemes as well as on abstract nodes, and so the word-word dependencies are only a subset of the dependencies in TigerDB. To make it comparable with the other dependency annotations, encoding dependencies between words, we only used the 1,567 sentences (29,373 tokens) of the TigerDB with lexically-rooted dependency structures and examined only dependencies between words, thus ignoring all abstract and sublexical nodes in the determination of the variation nuclei.[14]

The abstractness of the annotation of the TigerDB was already mentioned in the discussion of the differences between dependency annotation schemes in section 3.2, so let us be explicit about where it played a role for the error detection method. Firstly, the TigerDB itself does not directly encode the original lexical items, so we used the TigerDB token numbers to align each word-word dependency from the TigerDB with the surface string from the original Tiger treebank. Secondly, some words in the surface string are not annotated on the word level of the TigerDB. Some kinds of tokens are encoded as features (e.g., auxiliaries appear as tense features), others as abstract nodes (e.g., conjunctions), and some are not annotated at all (e.g., punctuation). However, since the variation detection method examines each dependency relation separately, it is not necessary for the dependency graph to be connected or for every word to be annotated. Unannotated words still play a role as context words, though, so that obtaining the surface form and order of the words from the original Tiger treebank, as we have done, is necessary.

Given the very small size of the corpus, the method detects only three variation nuclei when the non-fringe heuristic is used, i.e., when one word of identical context is required around the nucleus. All of the detected variation arises from annotation errors, but it clearly is crucial to explore alternate context requirements in order to improve recall. If no context is required at all, we obtain 1,060 variation nuclei. Applying only the NIL internal context heuristic discussed in section 4.2,

[14]Sublexical nodes occur in the TigerDB in the analysis of compound words. The final word in the compound is annotated as a lexical node and participates in word-word dependencies in the sentence, while the relationships within the compound are annotated using sublexical nodes (and thus are ignored by the error detection method).

the method detects 274 cases. We examined the cases and found that 48.5% (133) contain at least one error, corresponding to 149 token errors.

Looking at the nature of the errors detected for this corpus, of these 149 token errors, only 46 involved a labeling error, while 103 were an error in the identification of a dependency (of which 47 were incorrect labels and 56 incorrect NIL nuclei). Detecting twice as many dependency identification errors than dependency labeling errors contrasts with the results for the previous two corpora, where we found about the same number of dependency labeling and dependency identification errors (Talbanken05) or about 10% more labeling errors (PDT2.0). We interpret this as reflecting the fact that the TigerDB was semi-automatically constructed from another representational framework, LFG f-structure representations which were disambiguated using a broad-coverage LFG parser. Given the wider range of representational options, the identification of the lexical dependencies seems to have suffered.

In terms of a qualitative analysis of the errors detected, some common patterns can be observed in the annotation errors detected by our method. Firstly, consistent tokenization is a significant problem for multi-word expressions and proper names. For example, some instances of *Den Haag* ('The Hague') and *zur Zeit* ('at that time') are analyzed as single tokens while other instances of those expressions are annotated as several tokens. Secondly, the distinction between prepositional arguments and modifiers is a source of significant variation in the annotation. For example, in the expression *Bedarf an X* ('demand for X'), the prepositional phrase beginning with *an* was annotated as a modifier in one instance and as a prepositional object in another.

Turning to the false positives produced by the error detection method, we found genuine scope ambiguities with *nur* ('only') and *auch* ('also') and ambiguous words such as *die*, which can be an article, a relative pronoun, or a demonstrative pronoun. Such ambiguous words could be disambiguated based on their part-of-speech tags, which would significantly reduce the number of false positives.

Finally, we tested the effect of the dependency context heuristic. When applying that heuristic to the output of the NIL internal context heuristic, we find 89 variations with 64.0% precision. The drop in recall thus is significantly smaller than for the non-fringe heuristic applied to this corpus, with a clear increase in precision over the NIL internal context heuristic baseline. While this is the biggest gain in precision for the dependency context heuristic among the three treebanks, one needs to keep in mind that in the other two cases the dependency context heuristic only came into play after the non-fringe heuristic had already been applied.

6 Extensions and Alternatives

While the method successfully detects errors in a range of dependency treebanks, it is interesting to consider whether the recall of the method can be increased. A straightforward extension is to abstract from the word nuclei to distributional classes, i.e., POS tags (Boyd et al. 2007). While worth pursuing, this would require significant adaptations of the method in order to maintain adequate precision; Boyd et al. (2007), for example, develop three new heuristics in order to overcome the low error detection precision otherwise resulting from abstracting to distributional classes.

Another possibility for increasing recall would be to use more sophisticated machine learning methods and allow more features into the model, as opposed to using the actual words as nuclei and surrounding context words. The variation *n*-gram method is essentially a form of nearest neighbor classification, with a strict notion of neighbor (i.e., exact string match); relaxing the definition of a neighbor could lead to improved recall. In fact, there has been some work using a representation similar to ours for dependency parsing in a memory-based learning (MBL) framework: Canisius et al. (2006) treat dependency parsing as a classification problem where two words need to be mapped to a label, potentially one representing no relation (cf. our NIL label). While exploring the relation between the variation *n*-gram method and more general methods like MBL could be fruitful, it needs to be approached with caution, as the success of identifying inconsistencies relies on a data representation which predicts the same class with high precision. As mentioned above, the more general the representation, the more likely we will have low precision. van Halteren (2000), for example, only obtains 13-20% error detection precision by examining disagreements between a tagger and a benchmark corpus. Likewise, Blaheta (2002) finds 18% precision for function tag disagreements. To extend the variation *n*-gram method to more general nearest neighbor models, the essential work lies in selecting features which predict consistency without overflagging.

Other machine learning methods could also be attempted, such as outlier detection.[15] In addition to model selection, feature selection is again critical: for POS annotation, Eskin (2000) uses anomaly detection to find errors and has both lower precision and recall than the variation *n*-gram method on the same corpus. The kinds of errors that human annotators make—and which the variation *n*-gram method detects—tend to be errors which reflect a misunderstanding of the annotation scheme or are simply difficult to disambiguate; outlier detection seems less optimized for detecting such types of errors.

[15] Another perspective would be that of a noisy channel model; however, the lack of a training corpus with annotated errors (for each of the corpora considered here) makes it difficult to estimate the parameters of the error model.

7 Summary and Outlook

We developed a general, automatic method for detecting annotation errors in corpora with dependency annotation. Building on the variation *n*-gram method for constituency-based annotation (Dickinson and Meurers 2005a), we contrasted relevant properties of constituency-based and dependency-based annotation to define variation nuclei and disambiguating contexts which account for the properties of overlap and non-contiguity in dependency annotation. To the best of our knowledge the resulting method is the first general approach to detecting errors in dependency annotation. Experiments on three high-quality dependency treebanks demonstrate the effectiveness of the method and its ability to provide relevant feedback on the annotation schemes.

Given that the method relies upon inconsistently-annotated recurring data, it works best for large treebanks where the same words occur multiple times. While this is a potential limitation of the method, it is important to emphasize that the method successfully discovers cases which are difficult to distinguish or insufficiently documented. In other words, as mentioned before, the method points not only to specific errors, but to classes of errors, which annotators can use to guide the detection of additional errors, unclear annotation scheme decisions, or missing annotation manual documentation for difficult cases.

In terms of highlighting prominent patterns of inconsistency, a particular case caught by our constraints deserves attention here, namely variations involving NIL nuclei, i.e., cases where a pair of words has not been correctly identified as a dependency. The interpretation of a NIL nucleus for dependency annotation is quite different from that of constituency annotation: with constituency, a NIL nucleus might mean that a structure was simply flatter than it should have been. With dependency annotation, however, an erroneous NIL nucleus means not only that the pair of words are wrongly annotated, but that there is at least one other error in the sentence (assuming single-headedness): the word that should be the dependency is incorrectly the dependency of another word. Thus, investigating sentences with erroneous NIL nuclei will generally lead to the identification of multiple errors.

In future work it would be interesting to explore how one could generalize the notion of the variation nucleus and the disambiguating context to more abstract features, e.g., encoding a combination of the lemma and morphological information or part of speech, in order to increase the recall of the method (cf. Boyd et al. 2007). With such generalizations to both the nuclei and the contexts, we envisage that the method could also become applicable to related annotation levels, such as the predicate-argument structure and other semantic levels for which annotation efforts are becoming increasingly prominent (cf., e.g., Baker et al. 1998, Palmer et al. 2005).

Acknowledgments We want to thank the creators of the corpora for making them publicly available, especially Martin Forst for his help with TigerDB, and we want to thank Joakim Nivre, Mattias Nilsson, and Eva Pettersson for kind assistance with the evaluation of Talbanken05 and Jirka Hana and Jan Stepanek for the evaluation of PDT2.0. We are grateful for the support of this research by the National Science Foundation under Grant No. IIS-0623837.

References

Abeillé, Anne (ed.) (2003). *Treebanks: Building and using syntactically annotated corpora*. Dordrecht: Kluwer.

Baker, Collin F., Charles J. Fillmore and John B. Lowe (1998). The Berkeley FrameNet Project. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics (ACL/COLING)*. San Francisco, California, pp. 86–90.

Bick, Eckhard (2006). LingPars, a Linguistically Inspired, Language-Independent Machine Learner for Dependency Treebanks. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. New York City: Association for Computational Linguistics, pp. 171–175. http://aclweb.org/anthology/W06-2923.

Blaheta, Don (2002). Handling noisy training and testing data. In *Proceedings of the 7th conference on Empirical Methods in Natural Language Processing*. pp. 111–116. http://www.cs.brown.edu/~dpb/papers/dpb-emnlp02.html.

Boyd, Adriane, Markus Dickinson and Detmar Meurers (2007). Increasing the Recall of Corpus Annotation Error Detection. In *Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories (TLT 2007)*. Bergen, Norway.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius and George Smith (2002). The TIGER Treebank. In *Proceedings of TLT-02*. Sozopol, Bulgaria. http://www.bultreebank.org/proceedings/paper03.pdf.

Canisius, Sander, Toine Bogers, Antal van den Bosch, Jeroen Geertzen and Erik Tjong Kim Sang (2006). Dependency parsing by inference over high-recall dependency predictions. In *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X*. New York City, NY.

Carroll, John, Guido Minnen and Ted Briscoe (2003). Parser evaluation using a grammatical relation annotation scheme. In Abeillé (2003), chap. 17, pp. 299–316. http://treebank.linguist.jussieu.fr/pdf/17.pdf.

Dickinson, Markus (2005). Error detection and correction in annotated corpora. Ph.D. thesis, The Ohio State University. http://ling.osu.edu/~dickinso/papers/diss/.

Dickinson, Markus and W. Detmar Meurers (2003a). Detecting Errors in Part-of-Speech Annotation. In *Proceedings of EACL-03*. Budapest, pp. 107–114. http://ling.osu.edu/~dm/papers/dickinson-meurers-03.html.

Dickinson, Markus and W. Detmar Meurers (2003b). Detecting Inconsistencies in Treebanks. In *Proceedings of TLT-03*. Växjö, Sweden, pp. 45–56. http://ling.osu.edu/~dm/papers/dickinson-meurers-tlt03.html.

Dickinson, Markus and W. Detmar Meurers (2005a). Detecting Errors in Discontinuous Structural Annotation. In *Proceedings of ACL-05*. pp. 322–329. http://aclweb.org/anthology/P05-1040.

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 TLT-05*. Barcelona, Spain. http://ling.osu.edu/~dm/papers/dickinson-meurers-tlt05.html.

Einarsson, Jan (1976a). *Talbankens skriftsprøakskonkordans*. Tech. rep., Lund University, Dept. of Scandinavian Languages.

Einarsson, Jan (1976b). *Talbankens talsprøakskonkordans*. Tech. rep., Lund University, Dept. of Scandinavian Languages.

Eisner, Jason M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. Copenhagen, pp. 340–345. http://aclweb.org/anthology/C96-1058.

Eskin, Eleazar (2000). Automatic Corpus Correction with Anomaly Detection. In *Proceedings of NAACL-00*. Seattle, Washington. http://www.cs.columbia.edu/~eeskin/papers/treebank-anomaly-naacl00.ps/home/dm/resources/papers/eskin-00.ps.

Forst, Martin, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra and Valia Kordoni (2004). Towards a Dependency-Based Gold Standard for German Parsers. The TIGER Dependency Bank. In Silvia Hansen-Schirra, Stephan Oepen and Hans Uszkoreit (eds.), *Proceedings of LINC-04*. Geneva, Switzerland: COLING, pp. 31–38. http://aclweb.org/anthology/W04-1905.

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 EMNLP-CoNLL*. pp. 1084–1092.

Hajič, Jan, Alena Böhmová, Eva Hajičová and Barbora Vidová-Hladká (2003). The Prague Dependency Treebank: A Three-Level Annotation Scenario. In Abeillé (2003), chap. 7, pp. 103–127. http://ufal.mff.cuni.cz/pdt2.0/publications/HajicHajicovaAl2000.pdf.

Hajič, Jan, Barbora Vidová-Hladká and Petr Pajas (2001). The Prague Dependency Treebank: Annotation Structure and Support. In *Proceedings of the IRCS Workshop on Linguistic Databases*. University of Pennsylvania, Philadelphia, pp. 105–114. http://ufal.mff.cuni.cz/pdt2.0/publications/HajicHladkaPajas2001.pdf.

Hajič, Jan, Jarmila Panevová, Eva Buráňová, Zdeňka Urešová and Alla Bémová (1999). *Annotations at Analytical Layer. Instructions for Annotators*. Tech. rep., ÚFAL MFF UK, Prague, Czech Republic. English translation by Zdeněk Kirschner, http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/a-layer/pdf/a-man-en.pdf.

Hana, Jiří and Dan Zeman (2005). *A Manual for Morphological Annotation, 2nd edition*. Tech. Rep. 27, ÚFAL MFF UK, Prague, Czech Republic. http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/m-layer/pdf/m-man-en.pdf.

Havelka, Jiří (2007). Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 608–615.

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

Hudson, Richard A. (1990). *English Word Grammar*. Oxford: Blackwell.

King, Tracy H., Richard Crouch, Stefan Riezler, Mary Dalrymple and Ronald M. Kaplan (2003). The PARC 700 Dependency Bank. In *Proceedings of LINC-03*. Budapest. http://www2.parc.com/isl/groups/nltt/fsbank/.

Klein, Dan and Christopher D. Manning (2002). A Generative Constituent-Context Model for Improved Grammar Induction. In *Proceedings of ACL-02*. Philadelphia, PA.

Lin, Dekang (2003). Dependency-based evaluation of MINIPAR. In Abeillé (2003), chap. 18, pp. 317–332. http://www.cfilt.iitb.ac.in/archives/minipar_evaluation.pdf.

Manning, Christopher D. and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

McDonald, Ryan and Fernando Pereira (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Trento. http://aclweb.org/anthology/E06-1011.

Mel'čuk, Igor Aleksandrovic (1988). *Dependency syntax: theory and practice*. SUNY series in linguistics. Albany, NY: State University Press of New York.

Meurers, Detmar and Stefan Müller (2008). Corpora and Syntax (Article 44). In Anke Lüdeling and Merja Kytö (eds.), *Corpus Linguistics. An International Handbook*, Berlin: Mouton de Gruyter, Handbooks of Linguistics and Communication Science. http://ling.osu.edu/~dm/papers/meurers-mueller-07.html.

Mikulová, Marie, Allevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka et al. (2006). *Annotation on the Tectogrammatical Layer in the Prague Dependency Treebank. Annotation manual*. Tech. rep., ÚFAL MFF UK, Prague, Czech Republic. English translation, http://ufal.mff.cuni.cz/pdt2.0/doc/manuals/en/t-layer/pdf/t-man-en.pdf.

Mintz, T. H. (2003). Frequent frames as a cue for grammatical categories in child directed speech. *Cognition* 90, 91–117.

Mintz, T. H. (2006). Finding the verbs: distributional cues to categories available to young learners. In K. Hirsh-Pasek and R. M. Golinkoff (eds.), *Action Meets Word: How Children Learn Verbs*, New York: Oxford University Press, pp. 31–63.

Nilsson, Jens and Johan Hall (2005). *Reconstruction of the Swedish Treebank Talbanken*. MSI report 05067, Växjö University: School of Mathematics and Systems Engineering. http://w3.msi.vxu.se/~jni/papers/msi_report05067.pdf.

Nivre, Joakim (2005). *Dependency Grammar and Dependency Parsing*. MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Nivre, Joakim (2006). *Inductive Dependency Parsing*. Berlin: Springer.

Nivre, Joakim, Jens Nilsson and Johan Hall (2006). Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC)*. Genoa, Italy. http://www.vxu.se/msi/users/nivre/papers/talbanken05.pdf.

Padro, Lluis and Lluis Marquez (1998). On the Evaluation and Comparison of Taggers: the Effect of Noise in Testing Corpora. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics (ACL/COLING)*. San Francisco, California, pp. 997–1002. http://aclweb.org/anthology/P98-2164.

Palmer, Martha, Daniel Gildea and Paul Kingsbury (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics* 31(1), 71–105. http://aclweb.org/anthology/J05-1004.

Sgall, Petr, Eva Hajičová and Jarmila Panevová (1986). *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Prague, Czech Republic/Dordrecht, Netherlands: Academia/Reidel Publishing Company.

Tapanainen, Pasi and Timo Järvinen (1997). A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP)*. Washington, D.C. http://www.conexor.fi/anlp97/anlp97.ps/home/dm/resources/papers/tapanainen-jaervinen-anlp97.ps.

Taylor, Ann, Mitchell Marcus and Beatrice Santorini (2003). The Penn Treebank: An Overview. In Abeillé (2003), chap. 1, pp. 5–22.

Telljohann, Heike, Erhard W. Hinrichs, Sandra Kübler and Heike Zinsmeister (2005). *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Tech. rep., Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Tesnière, Lucien (1959). *Éléments de syntaxe structurale*. Paris: Editions Klincksieck.

Ule, Tylman and Kiril Simov (2004). Unexpected Productions May Well be Errors. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LCREC)*. Lisbon, Portugal. http://www.sfs.uni-tuebingen.de/~ule/Paper/us04lrec.pdf.

van Halteren, Hans (2000). The Detection of Inconsistency in Manually Tagged Text. In Anne Abeillé, Thorsten Brants and Hans Uszkoreit (eds.), *Proceedings of LINC-00*. Luxembourg. Workshop information at http://www.coli.uni-sb.de/linc2000/.

van Halteren, Hans, Walter Daelemans and Jakub Zavrel (2001). Improving Accuracy in Word Class Tagging through the Combination of Machine Learning Systems. *Computational Linguistics* 27(2), 199–229. http://aclweb.org/anthology/J01-2002.