Using Derivation Trees for Informative Treebank Inter-Annotator Agreement Evaluation

Seth Kulick and Ann Bies and Justin Mott and Mohamed Maamouri
Linguistic Data Consortium
University of Pennsylvania
{skulick,bies,jmott,maamouri}
@ldc.upenn.edu

Beatrice Santorini and Anthony Kroch
Department of Linguistics
University of Pennsylvania
{beatrice,kroch}
@ling.upenn.edu

Abstract

This paper discusses the extension of a system developed for automatic discovery of treebank annotation inconsistencies over an entire corpus to the particular case of evaluation of inter-annotator agreement. This system makes for a more informative IAA evaluation than other systems because it pinpoints the inconsistencies and groups them by their structural types. We evaluate the system on two corpora - (1) a corpus of English web text, and (2) a corpus of Modern British English.

1 Introduction

This paper discusses the extension of a system developed for automatic discovery of treebank annotation inconsistencies over an entire corpus to the particular case of evaluation of inter-annotator agreement (IAA). In IAA, two or more annotators annotate the same sentences, and a comparison identifies areas in which the annotators might need more training, or the annotation guidelines some refinement. Unlike other IAA evaluation systems, this system application results in a precise pinpointing of inconsistencies and the grouping of inconsistencies by their structural types, making for a more informative IAA evaluation.

Treebank annotation, consisting of syntactic structure with words as the terminals, is by its nature more complex and therefore more prone to error than many other annotation tasks. However, high annotation consistency is crucial to providing reliable training and testing data for parsers and linguistic research. Error detection is therefore an important area of research, and the importance of work such as Dickinson and Meurers (2003) is that errors and annotation inconsistencies might be automatically discovered, and once discovered, be targeted for subsequent quality control.

A recent approach to this problem (Kulick et al., 2011; Kulick et al., 2012) (which we will call the KBM system) improves upon Dickinson and Meurers (2003) by decomposing the full syntactic tree into smaller units, using ideas from Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997). This allows the comparison to be based on small syntactic units instead of string n-grams, improving the detection of inconsistent annotation.

The KBM system, like that of Dickinson and Meurers (2003) before it, is based on the notion of comparing identical strings. In the general case, this is a problematic assumption, since annotation inconsistencies are missed because of superficial word differences between strings which one would want to compare.[1] However, this limitation is not present for IAA evaluation, since the strings to compare are, by definition, identical.[2] The same is also true of parser evaluation, since the parser output and the gold standard are based on the same sentences.

We therefore take the logical step of applying the KBM system developed for automatic discovery of annotation inconsistency to the special case of IAA.[3]

[1] Boyd et al. (2007) and other current work tackles this problem. However, that is not the focus of this paper.

[2] Aside from possible tokenization differences by annotators.

[3] In this paper, we do not yet apply the system to parser evaluation, although it is conceptually the same problem as IAA evaluation. We wanted to first refine the system using annotator input for the IAA application before applying it to parser

Proceedings of NAACL-HLT 2013, pages 550–555,
Atlanta, Georgia, 9–14 June 2013. ©2013 Association for Computational Linguistics

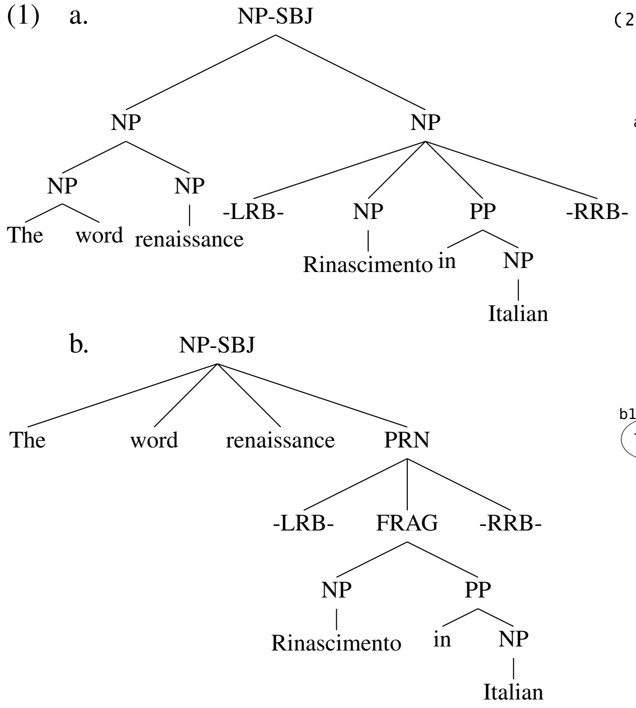


Figure 1: Two example trees showing a difference in IAA

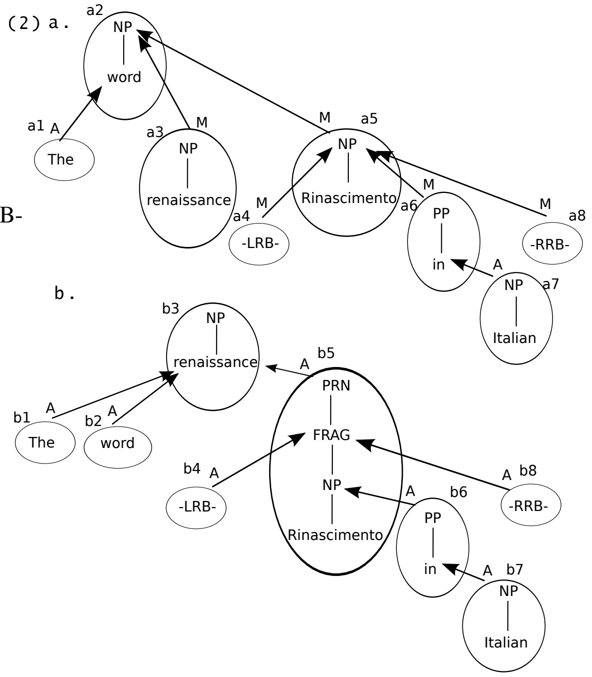


Figure 2: E-trees and derivation trees corresponding to (1ab)

To our knowledge, this work is the first to utilize such a general system for this special case.

The advantages of the KBM system play out somewhat differently in the context of IAA evaluation than in the more general case. In this context, the comparison of word sequences based on syntactic units allows for a precise pinpointing of differences. The system also retains the ability to group inconsistencies together by their structural type, which we have found to be useful for the more general case. Together, these two properties make for a useful and informative system for IAA evaluation.

In Section 2 we describe the basic working of our system. In Section 3 we discuss in more detail the advantages of this approach. In Section 4 we evaluate the system on two treebanks, a corpus of English web text and a corpus of Modern British English. Section 5 discusses future work.

2 System Overview

The basic idea of the KBM system is to detect word sequences that are annotated in inconsistent ways by comparing local syntactic units. Following Dickinson and Meurers (2003), we refer to sequences examined for inconsistent annotation as nuclei. The sentence excerpts (1ab) in Figure 1, from the test corpora used in this work, illustrate an inconsistency in the annotation of corresponding strings. We focus here on the difference in the annotation of the nucleus *The word renaissance*, which in (1a) is annotated as an appositive structure, while in (1b) it is flat.

Following the TAG approach, KBM decomposes the full phrase structure into smaller chunks called elementary trees (henceforth, e-trees). The relationship of the e-trees underlying a full phrase structure to each other is recorded in a derivation tree, in which each node is an e-tree, related to its parent node by a composition operation, as shown in (2ab).[4]

KBM uses two composition operations, each with left and right variants, shown in Figure 3: (1) ad-

evaluation.

[4] The decomposition is based on head-finding heuristics, with the result here that *word* is the head of (1a), while *renaissance* is the head of (1b), as reflected in their respective derivation trees (2a) and (2b). We omit the POS tags in (1ab) and (2ab) to avoid clutter.

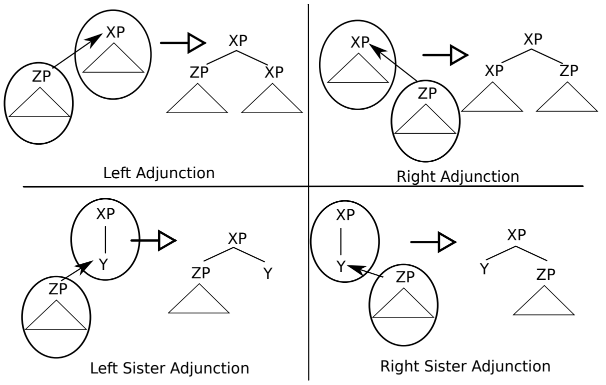


Figure 3: Composition operations (left and right)

junction, which attaches one tree to a target node in another tree by creating a copy of the target node, and (2) sister adjunction, which attaches one tree as a sister to a target node in another tree. Each arc in Figure 2 is labeled by an "M" for adjunction and "A" for sister-adjunction. [5]

The system uses the tree decomposition and resulting derivation tree for the comparison of different instances of the same nucleus. The full derivation tree for a sentence is not used, but rather only that slice of it having e-trees with words that are in the nucleus being examined, which we call a derivation tree fragment. That is, for a given nucleus with a set of instances, we compare the derivation fragments for each instance.

For example, for the nucleus *The word renaissance*, the derivation tree fragment for the instance in (1a) consists of the e-trees a1, a2, a3 (and their arcs) in (2a), and likewise the derivation tree from the instance in (1b) consists of the e-trees b1, b2, b3 in (2b). These derivation fragments have a different structure, and so the two instances of *The word renaissance* are recognized as inconsistent.

Two important aspects of the overall system require mention here: (1) Nuclei are identified by using sequences that occur as a constituent anywhere in the corpus, even if other instances of the same sequence are not constituents. Both instances of *The word renaissance* are compared, because the sequence occurs at least once as a constituent. (2) We partition each comparison of the instances of a nucleus by the lowest nonterminal in the derivation tree fragment that covers the sequence. The two instances of *The word renaissance* are compared because the lowest nonterminal is an NP in both instances.

3 Advantages of this approach

As Kulick et al. (2012) stressed, using derivation tree fragments allows the comparison to abstract away from interference by irrelevant modifiers, an issue with Dickinson and Meurers (2003). However, in the context of IAA, this advantage of KBM plays out in a different way, in that it allows for a precise pinpointing of the inconsistencies. For IAA, the concern is not whether an inconsistent annotation will be reported, since at some level higher in the tree every difference will be found, even if the context is the entire tree. KBM, however, will find the inconsistencies in a more informative way, for example reporting just *The word renaissance*, not some larger unit. Likewise, it reports *Rinascimento in Italian* as an inconsistently annotated sequence.[6]

A critical desirable property of KBM that carries over from the more general case is that it allows for different nuclei to be grouped together in the system's output if they have the same annotation inconsistency type. As in Kulick et al. (2011), each nucleus found to be inconsistent is categorized by an inconsistency type, which is simply the collection of different derivation tree fragments used for the comparison of its instances, including POS tags but not the words. For example, the inconsistency type of the nucleus *The word renaissance* in (1ab) is the pair of derivation tree fragments (a1,a2,a3) and (b1,b2,b3) from (2ab), with the POS tags. This nu-

[5] KBM is based on a variant of Spinal TAG (Shen et al., 2008), and uses sister adjunction without substitution. Space prohibits full discussion, but multiple adjunction to a single node (e.g., a4, a6, a8 to a5 in (2a)) does not create multiple levels of recursion, while a special specification handles the extra NP recursion for the apposition with a2, a3, and a5. For reasons of space, we also leave aside a precise comparison to Tree Insertion Grammar (Chiang, 2003) and Spinal TAG (Shen et al., 2008).

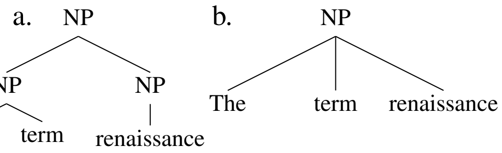
[6] Note however that it does *not* report *-LRB- Rinascimento in Italian -RRB-* which is also a constituent, and so might be expected to be compared. The lowest nonterminal above this substring in the two derivation trees in Figure 2 is the NP in a5 and the FRAG in b5, thus exempting them from comparison. It is exactly this sort of case that motivated the "external check" discussed in Kulick et al. (2012), which we have not yet implemented for IAA.

Inconsistency type	# Found	# Accurate
Function tags only	53	53
POS tags only	18	13
Structural	129	122

Table 1: Inconsistency types found for system evaluation

cleus is then reported together with other nuclei that use the same derivation fragments. In this case, it therefore also reports the nucleus *The term renaissance*, which appears elsewhere in the corpus with the two annotations from the different annotators as in (3):

(3)



KBM reports *The word renaissance* and *The term renaissance* together because they are inconsistently annotated in exactly the same way, in spite of the difference in words. This grouping together of inconsistencies based on structural characteristics of the inconsistency is critically important for understanding the nature of the annotation inconsistencies.

It is the combination of these two characteristics - (1) pinpointing of errors and (2) grouping by structure - that makes the system so useful for IAA. This is an improvement over alternatives such as using evalb (Sekine and Collins, 2008) for IAA. No other system to our knowledge groups inconsistencies by structural type, as KBM does. The use of the derivation tree fragments greatly lessens the multiple reporting of a single annotation difference, which is a difficulty for using evalb (Manning and Schuetze, 1999, p. 436) or Dickinson and Meurers (2003).

4 Evaluation

4.1 English web text

We applied our approach to pre-release subset of (Bies et al., 2012), dually annotated and used for annotator training, from which the examples in Sections 2 and 3 are taken. It is a small section of the corpus, with 4,270 words dually annotated.

For this work, we also took the further step of characterizing the inconsistency types themselves, allowing for an even higher-level view of the inconsistencies found. In addition to grouping together different strings as having the same inconsistent annotation, the types can also be grouped together for comparison at a higher level. For this IAA sample, we separated the inconsistency types into the three groups in Table 1, with the derivation tree fragments differing (1) only on function tags, (2) only on POS tags[7], and (3) on structural differences. We manually examined each inconsistency group to determine if it was an actual inconsistency found, or a spurious false positive. As shown in Table 1, the precision of the reported inconsistencies is very high. It is in fact even higher than it appears, because the seven (out of 129) instances incorrectly listed as structural problems were actually either POS or function tag inconsistencies, that were discovered by the system only by a difference in the derivation tree fragment, and so were categorized as structural problems instead of POS or function tag inconsistencies. [8]

Because of the small size of the corpus, there are relatively few nuclei grouped into inconsistency types. The 129 structural inconsistency types include 130 nuclei, with the only inconsistency type with more than one nucleus being the type with *The word renaissance* and *The term renaissance*, as discussed above. There is more grouping together in the "POS tags only" case (37 nuclei included in the 18 inconsistency types), and the "function tags only" case (56 nuclei included in the 53 inconsistency types).

4.2 Modern British English corpus

We also applied our approach to a supplemental section (Kroch and Santorini, in preparation) to a corpus of modern British English (Kroch et al., 2010), part of a series of corpora used for research into language change. The annotation style is similar to that of the Penn Treebank, although with some differences. In this case, because neither the function tags nor part-of-speech tags were part of the IAA work,

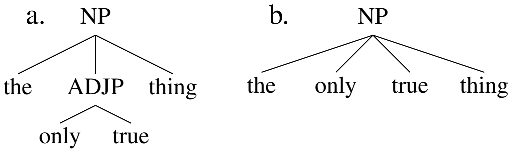
[7] As mentioned in footnote 4, although POS tags were left out of Figure 2 for readability, they are included in the actual e-trees. This allows POS differences in a similar syntactic context to be naturally captured within the overall KBM framework.

[8] A small percentage of inconsistencies are the result of linguistic ambiguities and not an error by one of the annotators.

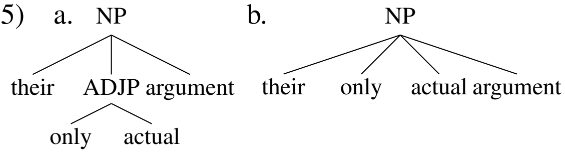
we do not separate out the inconsistency types, as done in Section 4.1.

The supplement section consisted of 82,701 words dually annotated. The larger size, as compared with the corpus in Section 4.1, results in some differences in the system output. Because of the larger size, there are more substantial cases of different nuclei grouped together as the same inconsistency type than in Section 4.1. The first inconsistency type (sorted by number of nuclei) has 88 nuclei, and the second has 37 nuclei. In total, there are 1,532 inconsistency types found, consisting of 2,194 nuclei in total. We manually examined the first 20 inconsistency types (sorted by number of nuclei), consisting in total of 375 nuclei. All were found to be true instances of inconsistent annotation.

(4)



(5)



The trees in (4) and (5) show two of the 88 nuclei grouped into the first inconsistency type. As with *The word renaissance* and *The term renaissance* in the English web corpus, nuclei with similar (although not identical) words are often grouped into the same inconsistency type. To repeat the point, this is not because of any search for similarity of the words in the nuclei. It arises from the fact that the nuclei are annotated inconstantly in the same way. Of course not all nuclei in an inconsistency type have the same words. Nuclei found in this inconsistency type include *only true* and *only actual* as shown above, and also nuclei such as *new English*, *greatest possible*, *thin square*, *only necessary*. Taken together, they clearly indicate an issue with the annotation of multi-word adjective phrases.[9]

5 Future work

There are several ways in which we plan to improve the current approach. As mentioned above, there is a certain class of inconsistencies which KBM will not pinpoint precisely, which requires adopting the "external check" from Kulick et al. (2012). The abstraction on inconsistency types described in Section 4 can also be taken further. For example, one might want to examine in particular inconsistency types that arise from PP attachment or that have to do with the PRN function tag.

One main area for future work is the application of this work to parser evaluation as well as IAA. For this area, there is some connection to the work of Goldberg and Elhadad (2010) and Dickinson (2010), which are both concerned with examining dependency structures of more than one edge. The connection is that those works are focused on dependency representations, and ithe KBM system does phrase structure analysis using a TAG-like derivation tree, which strongly resembles a dependency tree (Rambow and Joshi, 1997). There is much in this area of common concern that is worth examining further.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-11-C-0145. The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. This applies to the first four authors. The first, fifth, and sixth authors were supported in part by National Science Foundation Grant # BCS-114749. We would also like to thank Colin Warner, Aravind Joshi, Mitch Marcus, and the computational linguistics group at the University of Pennsylvania for helpful conversations and feedback.

[9] Note that the inconsistencies discussed throughout this paper are not taken from the the published corpora. These results are only from internal annotator training files.

References

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. LDC2012T13. Linguistic Data Consortium.

Adriane Boyd, 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.

David Chiang. 2003. Statistical parsing with an automatically extracted Tree Adjoining Grammar. In *Data Oriented Parsing*. CSLI.

Markus Dickinson and Detmar Meurers. 2003. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, Sweden. Treebanks and Linguistic Theories.

Markus Dickinson. 2010. Detecting errors in automatically-parsed dependency relations. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 729–738, Uppsala, Sweden, July. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. Inspecting the structural biases of dependency parsing algorithms. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 234–242, Uppsala, Sweden, July. Association for Computational Linguistics.

A.K. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 69–124. Springer, New York.

Anthony Kroch and Beatrice Santorini. in preparation. Supplement to the Penn Parsed Corpus of Modern British English.

Anthony Kroch, Beatrice Santorini, and Ariel Diertani. 2010. Penn Parsed Corpus of Modern British English. http://www.ling.upenn.edu/hist-corpora/PPCMBE-RELEASE-1/index.html.

Seth Kulick, Ann Bies, and Justin Mott. 2011. Using derivation trees for treebank error detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 693–698, Portland, Oregon, USA, June. Association for Computational Linguistics.

Seth Kulick, Ann Bies, and Justin Mott. 2012. Further developments in treebank error detection using derivation trees. In *LREC 2012: 8th International Conference on Language Resources and Evaluation*, Istanbul.

Christopher Manning and Hinrich Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In L. Wanner, editor, *Recent Trends in Meaning-Text Theory*, pages 167–190. John Benjamins, Amsterdam and Philadelphia.

Satoshi Sekine and Michael Collins. 2008. Evalb. http://nlp.cs.nyu.edu/evalb/.

Libin Shen, Lucas Champollion, and Aravind Joshi. 2008. LTAG-spinal and the Treebank: A new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.