Dangerous Relations in Dependency Treebanks

Chiara Alzetta[+], Felice Dell'Orletta[*], Simonetta Montemagni[*], Giulia Venturi[*]
[+] Università degli Studi di Genova
[*]Istituto di Linguistica Computazionale "A. Zampolli" (ILC–CNR), ItaliaNLP Lab
chiara.alzetta@edu.unige.it,
{felice.dellorletta,simonetta.montemagni,giulia.venturi}@ilc.cnr.it

Abstract

The paper illustrates an effective and innovative method for detecting erroneously annotated arcs in gold dependency treebanks based on an algorithm originally developed to measure the reliability of automatically produced dependency relations. The method permits to significantly restrict the error search space and, more importantly, to reliably identify patterns of systematic recurrent errors which represent dangerous evidence to a parser which tendentially will replicate them. Achieved results demonstrate effectiveness and reliability of the method.

1 Introduction

Dependency-based syntactic representations are playing more and more a key role in applications such as machine translation and information extraction (Kübler et al., 2009). If on the one hand current state-of-the-art approaches to dependency parsing require large training corpora, on the other hand dependency treebanks are very expensive to build in terms of both time and human effort.

The process of developing a treebank can be carried out in different ways, i.e. through: fully manual annotation; semi-automatic annotation, obtained via human editing of the automatic output of relevant NLP tools (e.g. POS taggers, dependency parsers); (semi-)automatic conversion from pre-existing resources. If fully manual annotation is time-consuming, costly and prone to inconsistencies even from a single annotator (Fort et al., 2012), semi-automatic annotation is faster, less prone to inconsistencies deriving from arbitrary decisions of the single annotator, but is subject to so-called "anchoring" effects according to which human decisions are affected by pre-existing values, which include parser errors (Berzak et al., 2016). More recently, available resources are more and more the result of a conversion process exploiting already existing annotated corpora: depending on whether conversion is carried out within the same syntactic representation paradigm, approaches can be *constituency-to-dependency* (Magerman, 1994; Yamada and Matsumoto, 2003; Nivre et al., 2006; Johansson and Nugues, 2007) or operate against dependency-based representations. Conversion can also be combined with merging and harmonization of different resources (Bosco et al., 2012): Nivre and Megyesi (2007) refers to this case as "cross-corpus harmonization". The conversion approach is particularly significant for less-resourced languages with limited annotated corpora or in the case of multi-lingual resources. The latter case is exemplified by the Universal Dependencies (UD) initiative (Nivre, 2015),[1] a recent community-driven effort to create cross-linguistically consistent dependency annotated corpora, where 70% of the released treebanks originate from a conversion process and only 29% of them has been manually revised after automatic conversion.

Whatever strategy is adopted for treebank construction, the resulting annotated corpus unavoidably contains errors. For this reason, the treebank annotation phase is usually followed by another step aimed at detecting and correcting errors. But treebank validation is as time-consuming as the annotation process: from this, the need follows for methods and techniques to support treebank validation by making the overall task fast and its result consistent and reliable. In principle, treebank validation is concerned

[1] http://universaldependencies.org

Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16), pages 201–210,
Prague, Czech Republic, January 23–24, 2018.

Keywords: error detection, Universal Dependency Treebanks, quality control of treebanks

with different types of errors. Following Agrawal et al. (2013), we distinguish: random errors, which are inherently unpredictable being typically due to annotators' distraction; errors connected with the annotation guidelines, due either to misinterpretation of the guidelines by the annotator, or to constructions not explicitly or comprehensively covered in the annotation guidelines and even errors in the provided guidelines, which are always evolving as long as annotation continues. To these, conversion errors should be added, i.e. errors due to either erroneous automatic mapping of an original annotation scheme to a new scheme or grey areas in the annotation of specific linguistic constructions. Whereas random errors are caused by unpredictable decisions by annotators, all other errors types can be classified as systematic and recurrent errors, that are not just determined by chance but are introduced by inaccuracies inherent to the procedure which generated them (automatic pre-annotation or conversion) or gaps in the annotation guidelines. In this paper, we will mainly focus on systematic and recurrent errors, which we qualify as "dangerous" for the fact of providing potentially "misleading" evidence to a parser during training, i.e. evidence leading to the replication of errors in the parser output.

In the literature, both pattern-based and statistical approaches have been adopted for carrying out error detection and correction in a rapid and reliable way. Relying on the intuition that "variation in annotation can indicate annotation errors", Dickinson and Meurers (2003, 2005) and Boyd et al. (2008) proposed a *variation n-gram* detection method where the source of variation is the so-called *variation nucleus*, i.e. "a word which has different taggings despite occurring in the same context, in this case surrounded by identical words". This methodology has been recently reimplemented and extended by de Marneffe et al. (2017) to detect inconsistencies in the UD treebanks. The idea that the cases where two "parsers predict dependencies different from the gold standard" are "the most likely candidates when looking for errors" was experimented by Volokh and Neumann (2011), who trained two parsers based on completely different parsing algorithms to reproduce the training data (i.e. the Penn Treebank). A similar pattern-based approach has been also proposed by Ambati et al. (2011) who complemented their method with a statistical module that, based on contextual features extracted from the Hindi treebank, was in charge of pruning previously identified candidate erroneous dependencies.

If all the aforementioned methods exploit corpus-internal evidence to detect inconsistencies within a given treebank, van Noord (2004) and de Kok et al. (2009) use external resources, i.e. they rely on the analysis of large automatically parsed corpora *external* to the treebank under validation. The underlying idea of these error mining techniques is that sentences with a low *parsability* score, i.e. sentences which have not received a successful analysis by the parser, very likely contain a parsing error.

This paper aims at testing the potential of algorithms developed to measure the reliability of automatically produced dependency relations for detecting erroneously annotated arcs in gold treebanks. In the literature, the result of this type of algorithms varies from a binary classification (correct vs. wrong) as in Che et al. (2014), to the ranking of dependencies on the basis of a quality score reflecting the reliability and plausibily of the automatic analysis (Dell'Orletta et al., 2013). Although these algorithms typically work on corpora automatically annotated (Dickinson, 2010), they have also been tested against corpora with manually revised (i.e. "gold") annotation: in this case, the typical aim is the identification of errors or simply inconsistencies in the annotation (Dickinson, 2015). In this work, we used an algorithm ranking dependencies by reliability, LISCA (Dell'Orletta et al., 2013), that was applied to a gold treebank to limit the search space for bootstrapping error patterns, i.e. systematic recurring errors (as opposed to random errors). Identified error patterns were then projected against the whole corpus. Like Ambati et al. (2011), here error detection is driven by statistical evidence which, in our approach, is acquired from an external automatically annotated large reference corpus.

2 Error Detection Methodology

The methodology devised to detect candidate errors in dependency treebanks is based on the parse quality assessment algorithm named LISCA (LInguiStically–driven Selection of Correct Arcs) (Dell'Orletta et al., 2013). As illustrated in details in Section 2.1, the algorithm exploits statistics about a wide range of linguistic features (covering different description levels, going from raw text to morpho-syntax and dependency syntax) extracted from a large reference corpus of automatically parsed sentences and uses

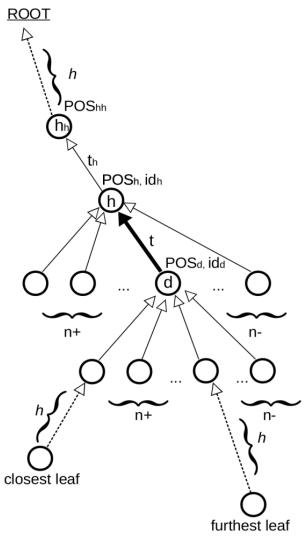


Figure 1: Features used by LISCA to measure arc(d, h, t) plausibility.

them to assign a *quality* score to each dependency arc contained in a target corpus belonging to the same variety of use (e.g. textual genre) of the automatically parsed corpus, thus producing a decreasing ranking of arcs from correct to anomalous ones, potentially including incorrect ones. The underlying assumption is that syntactic structures that are more frequently generated by a parser are more likely to be correct than less frequently generated structures.

2.1 The LISCA Algorithm

LISCA takes as input a set of parsed sentences and it assigns a plausibility score to each dependency, which is defined as a triple (d, h, t) where d is the dependent, h is the head, and t is the type of dependency connecting d to h. The algorithm operates in two steps: 1) it collects statistics about a set of linguistically motivated features extracted from a dependency annotated corpus obtained through automatic dependency parsing, and 2) it combines the feature statistics extracted from the corpus used during the previous step. The final plausibility score associated with a given dependency arc results from the combination of the weights associated with these features: the score is computed as a simple product of the individual feature weights.[2]

Figure 1 summarizes the features taken into account by LISCA for measuring the plausibility of a given syntactic dependency (d, h, t). For the purposes of the present study, LISCA has been used in its de–lexicalized version in order to abstract away from variation resulting from lexical effects. In particular, two different types of features are considered: *local* features, corresponding to the characteristics of the syntactic arc considered (e.g. the distance in terms of tokens between d and h, or the associative strength linking the grammatical categories, i.e. POS$_d$ and POS$_h$, involved in the relation, or the POS of the head governor and the type of syntactic dependency connecting it to h); *global* features, aimed at locating the arc being considered within the overall syntactic structure of the sentence, with respect to both its hierarchical structure and the linear ordering of words (for example, the distance of d from the root of the tree, or from the closest or most distant leaf node, or the number of "siblings" and "children" nodes of d, recurring respectively to its right or left in the linear order of the sentence).

LISCA was successfully used against both the output of dependency parsers and gold treebanks. While in the first case the plausibility score was meant to identify unreliable automatically produced dependency relations, in the second case it was used to detect shades of syntactic markedness of syntactic constructions in manually annotated corpora. The latter is the case of Tusa et al. (2016), where the LISCA ranking was used to investigate the linguistic notion of "markedness" (Haspelmath, 2016): a given linguistic construction is considered "marked" when it deviates from the "linguistic norm", i.e. it

[2]For a detailed description of the features and the metrics used by LISCA see Dell'Orletta et al. (2013).

is “abnormal”. Accordingly, unmarked constructions are expected to be characterized by higher LISCA scores and – conversely – constructions characterized by increasing degrees of markedness are associated with lower scores. In the analysis of their linguistic results, Tusa et al. (2016) noticed that low scored relations also included annotation errors. This observation prompted our hypothesis of research, i.e. that the identification of problematic areas of human annotation can be carried out by measuring the *distance* of the linguistic context characterizing the arcs in a gold treebank from the “linguistic norm” computed by LISCA with respect to a large reference corpus.

2.2 Chasing errors with LISCA

According to these premises, errors in gold treebanks were searched for with LISCA assuming that a higher number of *variations* of the linguistic context for an arc in the manual annotation with respect to the automatically generated arcs corresponds to a greater chance for the observed variation to be an error. In this respect, arc *variation* is observed whenever the linguistic context of an arc in the treebank differs with respect to the corresponding one captured in the large reference corpus used to compute the LISCA score. Similarly to Ambati et al. (2011), we exploited the contextual features of an arc to identify erroneous annotations but differently from them we looked for these features *outside* the treebank under analysis, thus overcoming the widely ackowledged data sparsity problem. By doing so, the error search space is restricted to relations with lower LISCA scores.

The proposed error detection method is articulated into the following steps:

1. LISCA is run against the gold treebank and arcs are ordered by decreasing LISCA scores;
2. the resulting ranking of arcs is partitioned it into 10 groups, henceforth “bins”, each corresponding to 10% of the total (plus an 11th bin for the remaining ones);
3. the analysis was limited to the last three bins containing relations associated with the lowest LISCA scores: these bins were expected to gather a higher occurrence of “abnormal” annotations, be they errors or less frequent constructions;
4. the selected bins were manually inspected to identify errors, both random errors and systematic errors (i.e. “dangerous relations”);
5. recurring systematic errors which emerged from this manual inspection were formalized as error patterns which were then projected onto the whole treebank;
6. potentially erroneous identified arcs in all bins were manually validated and - whenever needed - corrected.

Let us exemplify how the decreasing LISCA scores assigned to different instances of the same relation occurring within different linguistic contexts can be used to guide error detection.

	B1	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**B8**	**B9**	**B10**
Total occurrences	785	543	449	353	333	168	132	97	106	114
Errors (occurrences)	0	0	0	1	6	5	12	7	9	4
Errors (percentages)	0	0	0	0.28	1.80	2.97	9.09	7.21	8.49	3.51

Table 1: Occurrences of *mark* relation in the IUDT newspaper section and erroneously annotated instances across the LISCA bins.

Table 1 reports the distribution of the UD *mark* relation (linking the function word introducing a subordinated clause to the verbal head of the clause) across the LISCA bins in the newspaper section of the Italian Universal Dependency Treebank (the gold treebank we used to test our methodology, as described in Section 3). Although the relation occurs in all bins, the frequency of occurrence decreases proportionally to the decreasing of the scores assigned by LISCA. The higher frequency of the *mark* relation in

the top LISCA bins can be explained by the generally fixed or slightly variable structure underlying it: these occurrences correspond to canonical linguistic contexts which are closer to the "linguistic norm" as computed by LISCA with respect to the large reference corpus. By contrast, anomalous *mark* structures ended up in the last bins, in particular in the 7th-9th bins, for which a higher percentage of errors is reported (ranging between 7% and 9%).

3 Corpora

The proposed error detection methodology was tested against the Italian Universal Dependency Treebank (henceforth IUDT) (Bosco et al., 2013), which contains 13,815 sentences corresponding to 325,816 tokens. As de Marneffe et al. (2017) pointed out, UD treebanks represent a good testing bed for error detection techniques: most part of them originate from a conversion process, often combined with merging and cross-corpus harmonization. In particular, IUDT results from the harmonization and merging of smaller dependency–based resources adopting incompatible annotation schemes into the Universal Dependencies annotation formalism, with the final aim of constructing a standard-compliant and bigger resource for the Italian language: the Turin University Treebank (TUT, Bosco et al. (2000)) and ISST–TANL (originating from the ISST corpus, (Montemagni et al., 2003)).

For the specific concerns of this study, we focused on the section of IUDT containing newspaper articles, composed by 10,891 sentences, for a total of 154,784 tokens. This choice was aimed at avoiding possible interferences in detecting anomalies due to textual genre variation: in this case, "abnormal" relations do not only include possible errors but also constructions peculiar to a specific genre.

The corpus used to collect the statistics to build the LISCA model is represented by the *La Repubblica* corpus, a collection of newspaper articles part of the CLIC-ILC Corpus (Marinelli et al., 2003) for a total of 1,104,237 sentences (22,830,739 tokens). The corpus was morpho-syntactically annotated and parsed by the UDPipe pipeline (Straka et al., 2016) trained on IUDT, version 2.0 (Nivre et al., 2017).

4 Results

LISCA was used to rank the journalistic section of IUDT: the ranked relations were partitioned into 10 bins of about 14,600 arcs each, with an 11th bin with the remaining 8723 arcs. The manual revision focused on the last three bins (from 9th to 11th), covering 24.5% of the total number of arcs.

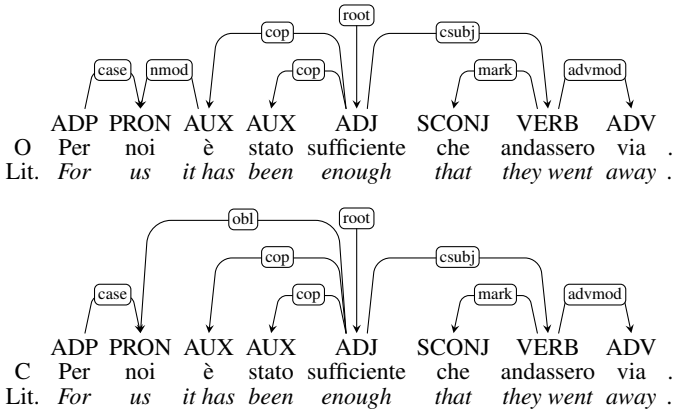
At the end of the error detection and correction process, 789 arcs were modified, corresponding to 0.51% of the number of arcs in IUDT news, distributed into 567 sentences (i.e. 5.21% of the number of sentences in IUDT news). Of those 789 arcs, 286 arcs (36.01%) are random errors: interestingly, 185 of them (i.e. 65% of random errors) are located in the 11th LISCA bin. The remaining detected errors, i.e. 503 (63.99%), represent systematic errors which have been identified on the basis of error patterns manually identified in the last bins and which have then been projected back onto the whole IUDT news section. These error patterns turned out to represent real errors in 85.63% of the cases, involving 483 sentences: this demonstrates the effectiveness of identified potential error patters.

4.1 Typology of Dangerous Relations

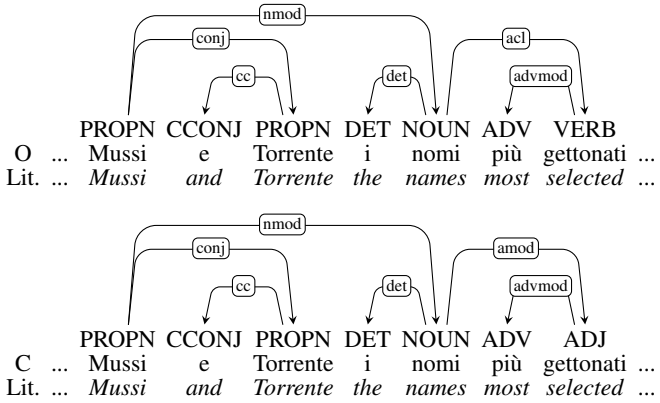
In what follows, we will illustrate the main systematic errors, corresponding to so–called "dangerous relations", which emerged from the analysis of relations in the last three bins and which were formalized as the following six error patterns.[3]

Auxiliary verbs (*aux_head*): it refers to cases where an auxiliary verb (i.e. *essere* 'to be', *avere* 'to have', modals, periphrastic or copular verbs) was erroneously treated as the head of a dependency relation, as in the following example where the personal pronoun *noi* 'us' was erroneously governed by the auxiliary verb *è* rather than by *sufficiente*, which represents the nonverbal predicate and root of the sentence:

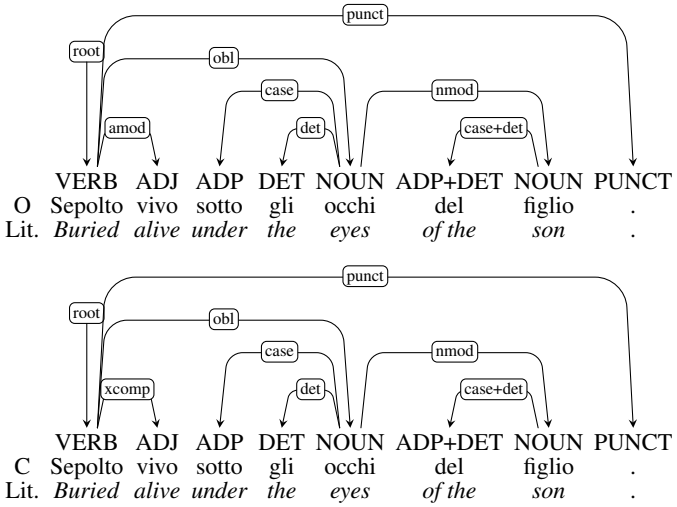
[3] In the following examples the original wrong sentence is marked with *O* (*Original*), and the corrected one is marked with *C* (*Correct*)



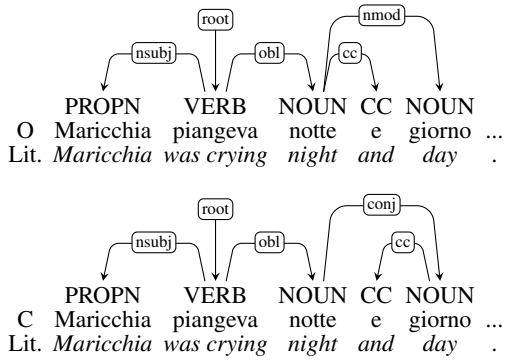
Clausal modifier of a noun (*acl4amod*): it refers to cases where bare past participles functioning as adjectival modifiers of nouns were erroneously annotated as clausal modifiers (i.e. *acl*). In these cases, the lemma, the part of speech and the type of dependency were modified, as in the following example where the past participle *gettonati* 'selected' was erroneously *i)* associated with the lemma *gettonare* 'to select' instead of the lemma *gettonato* 'selected', *ii)* morpho-syntactically tagged as VERB rather than ADJ, and *iii)* linked to the head word *nomi* 'names' with the relation *acl* rather than *amod*:



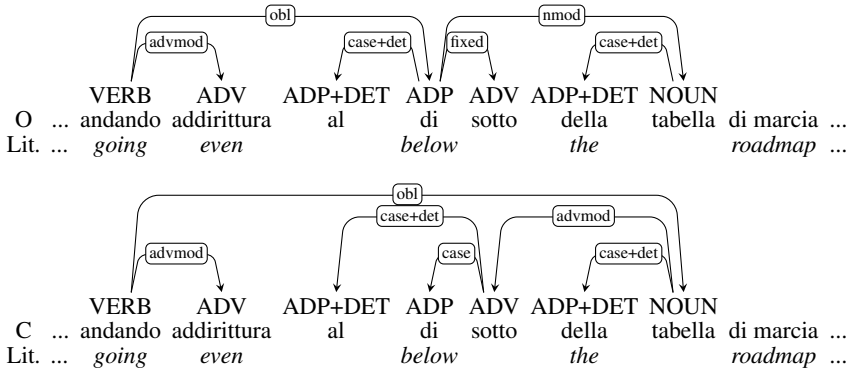
Adjectival modifiers (*amod4xcomp*): it refers to cases where adjectives functioning as secondary predicates of a verb were erroneously annotated as *amod* rather than *xcomp*, as in the following example where the syntactic function of adjectival modifier (*amod*) holding between the adjective *vivo* 'alive' and the head verb *sepolto* 'buried' was erroneously identified:



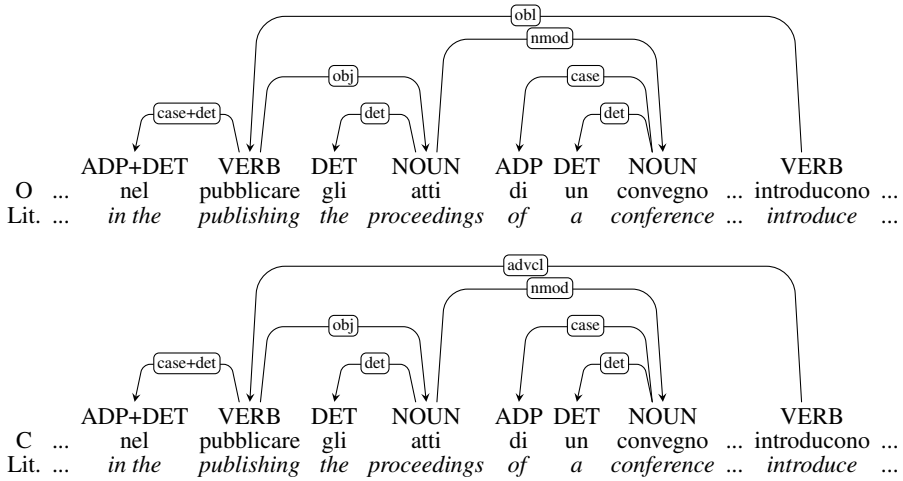
Coordinating conjunctions (*conj_head*): it refers to cases where a coordinating conjunction was erroneously headed by the first conjunct (coordination head), as in the following example where the conjunction *e* 'and' was headed by *notte* 'night' rather than by *giorno* 'day':



Nominal modifiers (*nmod4obl*): it refers to cases where an oblique argument was erroneously annotated as nominal modifier (*nmod*) rather than as oblique nominal (*obl*) when occurring in multiword expressions which were not correctly identified, as in the following example where the noun *tabella* 'chart' was erroneously headed by the preposition *di* 'of' rather than by the verb *andando* 'going', and linked by the dependency relation *nmod* rather than *obl*:



Nonfinite verbs (*obl4advcl|acl*): it refers to cases where nonfinite verbal constructions functioning as nominals were erroneously annotated as oblique nominals (*obl*) rather than adverbial or adjectival clauses (*advcl* or *acl*), as in the following example represented by the verb *pubblicare* 'publish':



4.2 Discussion

The patterns illustrated above can be classified under three main categories: 1) head identification errors (*aux_head*, *conj_head*), 2) labeling errors (*acl4amod*, *amod4xcomp*, *obl4advcl|acl*), and 3) combined head identification and labeling errors (*nmod4obl*). Table 2 shows the detail of the modified arcs for each pattern, while Figure 2 visualizes their distribution across the LISCA bins. The chart confirms the hypothesis we started from, i.e. that most part of systematic errors are concentrated in the last bins and that, on the other hand, the first LISCA bins tendentially do not contain errors, or very few of them.

Note that the 11th bin is not included in the chart since it turned out to only contain random errors (as opposed to systematic ones). If we try to track the origin of the identified and corrected recurrent errors, it is worth noting that the most frequent error type recorded in Table 2 – *acl4amod* – corresponds to a quite problematic annotation area for all treebanks, i.e. the distinction between participial and adjectival usages. More interestingly, this corresponds to an area for which the original resources which were combined in IUDT (i.e. TUT and ISST–TANL) followed different guidelines: for TUT, the verbal reading was preferred, which naturally led to the interpretation of (reduced) relative clause, whereas ISST–TANL resorted in these cases to a general modifier relation. The second and third most frequent errors (namely, *conj_head* and *aux_head*) are connected with substantial changes from version 1.4 to 2.0 of Universal Dependencies annotation guidelines. Last but not least, the error types *amod4xcomp* and *nmod4obl* seem rather to be connected to annotation inconsistencies internal to the treebank.

Error pattern	**Frequency**
Auxiliary verbs (aux_head)	13.32 (67)
Clausal modifiers of noun (acl4amod)	36.98 (186)
Adjectival modifiers (amod4xcomp)	12.52 (63)
Coordinating conjunctions (conj_head)	24.65 (124)
Nominal modifiers (nmod4obl)	6.76 (34)
Nonfinite verbs (obl4advcl|acl)	5.77 (29)
Total number of errors:	**503**

Table 2: Distribution (percentage and absolute values) of error types in IUDT.

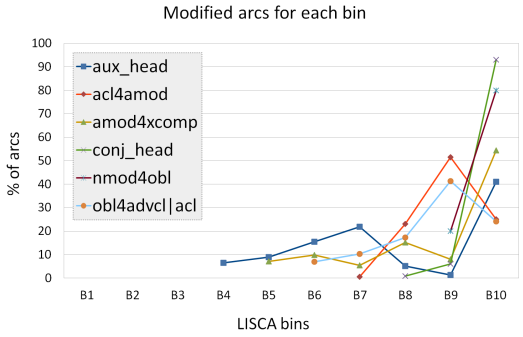


Figure 2: Distribution of modified arcs for each error pattern across the LISCA bins.

5 Conclusion and Current Directions of Research

We proposed an effective and innovative method for detecting erroneously annotated arcs in gold treebanks based on an algorithm originally developed to measure the reliability of automatically produced dependency relations, LISCA. This method permits to significantly restrict the error search space and, more importantly, to reliably identify patterns of systematic recurrent errors which represent dangerous and misleading evidence to a parser. Achieved results demonstrate the effectiveness of the method. Within the whole amount of corrected errors (both random and systematic), 64% corresponds to systematic errors, typically originating from semi-automatic annotation or conversion. The effectiveness of identified patterns is demonstrated by the fact that in the whole IUDT news section 85.67% of the sentences instantiating at least one error pattern contains real errors. In principle, this method, operating within the dependency-based representation framework, is independent from language and annotation scheme. As a preliminary experiment in this direction, we checked the presence of some detected error patterns (i.e. those due to problematic annotation areas and guidelines changes between different treebank versions) in other UD treebanks.[4] We looked for sentences instantiating the constructions corresponding to our error patterns in different UD treebanks: patterns turned out to appear both in languages typologically close to Italian (e.g. French, Spanish and Portuguese) and typologically distant (e.g. English, Arabic, Czech, Finnish, Turkish and Chinese). For most of those languages, the total number of sentences containing the patterns is in line with the number of sentences we found for Italian (between 3-7% over the total number of sentences in the treebank), with the exception of Turkish and Chinese where the number is much higher (around 15%). This holds true also for the distribution of patters: like for Italian, the most frequent pattern observed in all UD treebanks taken into account is *acl4amod*. This very preliminary evidence extracted from different UD treebanks needs however to be validated through a collaboration between different UD national teams, to assess whether identified anomalous patterns represent real errors. Current developments also include the assessment of impact and role of detected and corrected errors in the performance of dependency parsers.

[4]For this purpose we used the Dep_search tool developed by the Turku NLP Group.

Acknowledgments

The work reported in the paper was partially supported by the 2–year project (2016-2018) *Smart News, Social sensing for breaking news*, funded by Regione Toscana (BANDO FAR-FAS 2014). Thanks are also due to the University of Pisa who funded a post-graduate fellowship with a Google gift.

References

B. Agrawal, R. Agarwal, S. Husain, and D.M. Sharma. 2013. *An Automatic Approach to Treebank Error Detection Using a Dependency Parser*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 294–303.

B. R. Ambati, R. Agarwal, M. Gupta, S. Husain, and D. M. Sharma. 2011. Error Detection for Treebank Validation. In *Proceedings of 9th International Workshop on Asian Language Resources (ALR)*.

Y. Berzak, Y. Huang, A. Barbu, A. Korhonen, and B. Katz. 2016. Anchoring and Agreement in Syntactic Annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2215–2224.

C. Bosco, V. Lombardo, L. Lesmo, and D. Vassallo. 2000. Building a Treebank for Italian: a Data-driven Annotation Schema. In *Proceedings of the 2nd Language Resources and Evaluation Conference (LREC'00)*. Athens, Greece, pages 99–105.

C. Bosco, S. Montemagni, and M. Simi. 2012. Harmonization and Merging of two Italian Dependency Treebanks. In *Proceedings of the LREC 2012 Workshop on Language Resource Merging*. Istanbul, Turkey.

C. Bosco, S. Montemagni, and M. Simi. 2013. Converting Italian Treebanks: Towards an Italian Stanford Dependency Treebank. In *Proceedings of the ACL Linguistic Annotation Workshop & Interoperability with Discourse*. Sofia, Bulgaria.

A. Boyd, M. Dickinson, and W. D. Meurers. 2008. On Detecting Errors in Dependency Treebanks. *Research on Language & Computation* 6(2):113–137.

W. Che, J.Guo, and T.Liu. 2014. Reliable Dependency Arc Recognition. *Expert Systems with Applications* 41(4):1716–1722.

D. de Kok, J. Ma, and G. van Noord. 2009. A Generalized Method for Iterative Error Mining in Parsing Results. In *Proceedings Workshop on Grammar Engineering Across Frameworks (GEAF 2009)*.

M.C. de Marneffe, M. Grioni, J. Kanerva, and F. Ginter. 2017. Assessing the Annotation Consistency of the Universal Dependencies Corpora. In *Proceedings of the 4th International Conference on Dependency Linguistics (Depling 2007)*. Pisa, Italy, pages 108–115.

F. Dell'Orletta, G. Venturi G., and S. Montemagni. 2013. Linguistically-driven Selection of Correct Arcs for Dependency Parsing. *Computaciòn y Sistemas* 2:125–136.

M. Dickinson. 2010. Detecting Errors in Automatically-Parsed Dependency Relations. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, pages 729–738.

M. Dickinson. 2015. Detection of Annotation Errors in Corpora. *Language and Linguistics Compass* 9(3):119–138.

M. Dickinson and W. D. Meurers. 2003. Detecting Inconsistencies in Treebank. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*.

M. Dickinson and W. D. Meurers. 2005. Detecting Errors in Discontinuous Structural Annotation. In *Proceedings of the 43rd Annual Meeting of the ACL*. pages 322–329.

K. Fort, A. Nazarenko, and S. Rosset. 2012. Modeling the Complexity of Manual Annotation Tasks: a Grid of Analysis. In *Proceedings of COLING 2012*. pages 895–910.

M. Haspelmath. 2016. Against Markedness (and what to Replace it with). *Journal of Linguistics* 42:25–70.

R. Johansson and P. Nugues. 2007. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of NODALIDA 2007*.

S. Kübler, R. McDonald, and J. Nivre. 2009. *Dependency Parsing. Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

D. M. Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

R. Marinelli, L. Biagini, R. Bindi, S. Goggi, M. Monachini, P. Orsolini, E. Picchi, S. Rossi, N. Calzolari, and A. Zampolli. 2003. The Italian PAROLE Corpus: an Overview. *Linguistica Computazionale* XVIXVII:401–421.

S. Montemagni, F. Barsotti, M. Battista, N. Calzolari, A. Lenci, O. Corazzari, A. Zampolli, F. Fanciulli, M. Massetani, R. Basili, R. Raffaelli, M.T. Pazienza, D. Saracino, F. Zanzotto, F. Pianesi, N. Mana, and R. Delmonte. 2003. Building the Italian Syntactic-Semantic Treebank. In Anne Abeillé, editor, *Treebanks. Building and Using Parsed Corpora*, Springer Science Business Media, LLCs, pages 189–210.

J. Nivre. 2015. Towards a Universal Grammar for Natural Language Processing. In *Computational Linguistics and Intelligent Text Processing - Proceedings of the 16th International Conference, CICLing 2015, Part I*. Cairo, Egypt, pages 3–16.

J. Nivre, J. Hall, and J. Nilsson. 2006. Maltparser: A Data-driven Parser Generator for Dependency Parsing. In *Proceedings of LREC2006*.

J. Nivre and B. Megyesi. 2007. Bootstrapping a Swedish Treebank Using Cross-Corpus Harmonization and Annotation Projection. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories (TLT)*. pages 97–102.

J. Nivre, A. Željko, and A. Lars et al. 2017. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

M. Straka, J. Hajic, and J. Strakova. 2016. UD-Pipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*.

E. Tusa, F. Dell'Orletta, S. Montemagni, and G. Venturi. 2016. Dieci sfumature di marcatezza sintattica: verso una nozione computazionale di complessitá. In *Proceedings of the Third Italian Conference on Computational Linguistics (CLiC-it)*. Napoli, Italy, pages 3–16.

G. van Noord. 2004. Error Mining for Widecoverage Grammar Engineering. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

A. Volokh and G. Neumann. 2011. Automatic Detection and Correction of Errors in Dependency Treebanks. In *Proceedings of ACL-HLT (2011)*.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of 8th International Workshop on Parsing Technologies*.