ERRATOR: a Tool to Help Detect Annotation Errors in the Universal Dependencies Project

Guillaume Wisniewski
LIMSI, CNRS, Univ. Paris-Sud, Université Paris Saclay, 91 403 Orsay, France
guillaume.wisniewski@limsi.fr

Abstract

Enforcing guidelines compliance is today one of the main challenge faced by the Universal Dependencies project. This work introduces ERRATOR, a set of tools implementing the *annotation variation principle* that can be used to help annotators find and correct errors in the different layers of annotations of UD treebanks. The results of a first annotation campaign that used ERRATOR to correct and harmonize the annotations of the different French corpora are also described.

Keywords: Universal Dependencies, Annotation Coherence, Evaluation

1. Introduction

The Universal Dependencies project (Nivre et al., 2017) aims at developing cross-linguistically consistent treebank annotations for a wide array of languages. Each treebank contains raw text, sentence and word segmentation, PoS tags, dependency relations and in many cases lemmas and morphological features.

In its latest release, the UD project gathers 70 treebanks covering 50 languages. Many of these corpora result from a manual or semi-automatic transformation from existing dependency or constituent treebanks into the UD formalism (see, for instance, (Bosco et al., 2013) or (Lipenkova and Souček, 2014) for a description of such transformations). Because many treebanks have been annotated and/or converted independently by different groups, the risk of incoherence and errors in the application of annotation guidelines is increased. There may indeed be several sources of errors in the produced annotations: in addition to the divergences in the theoretical linguistic principles that governed the design of the original annotation guidelines, errors may also result from automatic (pre-)processes, human post-editing, or human annotation.

As a matter of fact, several works have recently pointed out that different treebanks for the same language are not consistently annotated even though they should follow the same guidelines (Aufrant et al., 2017; Vilares and Gómez-Rodríguez, 2017). More generally, (Wisniewski et al., 2014) has shown that, in spite of common annotation guidelines, the main bottleneck in cross-lingual transfer is the difference in the annotation conventions across corpora and languages.

Enforcing guidelines compliance to improve annotation quality is therefore one of the main challenge faced by the UD project today: as it is, performance achieved on UD corpora, especially in a cross-lingual or cross-corpus setting may be underestimated and this drop may results mainly from divergences in annotations. This work describes ERRATOR, a set of tools implementing the *annotation variation principle* that has been proposed to detect errors in PoS annotations (van Halteren, 2000; Dickinson and Meurers, 2003) and syntactic annotations (Dickinson and Meurers, 2005; Boyd et al., 2008). We propose an extension of this principle to word segmentation making ERRATOR able to help annotators find and correct errors in the different layers of annotations of the UD corpora.

The rest of this paper is organized as follows: we will first explain how the annotation variation principle can be used to identify potential annotation errors (§ 2.). We will then describe our implementation (§ 3.) and the results of a first annotation campaign that used ERRATOR to correct and harmonize the annotations of the different French corpora. ERRATOR is open-source and can be freely downloaded from `https://perso.limsi.fr/wisniews/errator/`.

2. Identifying Potential Annotation Errors

Principle ERRATOR implements the *annotation variation principle* (Boyd et al., 2008) to detect potential annotation errors. This principle states that if two identical sequences of words are annotated in different ways, there is a high chance that one of the annotation is erroneous.

More precisely, we consider that a corpus annotated with PoS, tokenization or dependencies information defines an *alignment* between source sentences and their annotations (in the usual Machine Translation meaning). Following these alignment links, formally defined in the next paragraph, it is possible, given a substring of a source sentence to extract the 'corresponding' part of the annotation. The annotation variation principle can then be implemented as follows:

1. Given one or several corpora of annotated sentences, find all *maximal repeats*, that is to say a substring that occurs at least in two different sentences and cannot be extended to the left or to right to a longer substring. The *maximal repeat problem* can be solved efficiently[1] using Generalized Suffix Tree (Gusfield, 1997).
2. For all repeats, extract their corresponding annotation following alignment links;
3. if not all the annotations corresponding to a single repeat are identical, flag them as a potential annotation error.

[1] If the corpus contain n words, extracting all the maximal repeats takes $\mathcal{O}(n)$ to build the GST and $\mathcal{O}(n)$ to list all the repeats.

These potential errors have then to be reviewed by a human annotator that can eventually correct them.

Aligning source text with their annotation As explained in the previous paragraph, alignment aims at allowing us to 'extract' the part of the annotation that corresponds to a sub-part of a sentence. Aligning tokenized words with PoS or morphological information is straightforward as each word is associated to exactly one label.
For dependencies we represent the dependency tree with a list of pairs (head index, label),[2] that, again, can be mapped directly with the words of the sentence. To test if the dependency annotation of two identical (sub-)sequences of words are the same, we consider that heads outside the sequence are replaced by a special symbol and head indices are replaced by their relative position. Figure 1 shows an example of this representation.[3]
For word segmentation, things are more complicated as, as illustrated in Table 1, the tokenization is not necessarily concatenative: for instance, according to UD guidelines, concatenation and clitics are expanded. It is therefore necessary to define an alignment between the characters of the raw and segmented texts in which some of characters of the raw string can be aligned to more than one characters of the tokenized string.
The alignment is built as follows: we start by looking for the longest common substring (at the character level) between the raw and the tokenized strings and align all their characters. We then remove this substring from both strings and, recursively, re-apply this procedure to the resulting strings as long as there is a common substring. Parts of the annotation that have not been matched are then arbitrarily aligned with the first character of the following match.

①	Souvent␣,␣celui-ci␣l'assume␣seul␣. Souvent␣,␣celui␣-␣ci␣l'␣assume␣seul␣.
②	je␣vais␣au␣jardin je␣vais␣à␣le␣jardin
③	la␣galerie␣des␣batailles␣,␣dans la␣galerie␣de␣les␣batailles␣,␣dans

Table 1: Example of the alignments for word segmentation: groups represented in the same color define 1:1 or 1:n alignments between characters. In examples ② and ③, the segmentation is not concatenative.

3. ERRATOR

In this Section, we describe ERRATOR a set of tools we have developed to automatically detect potential annotation errors and help users to filter out false positive and correct true annotation errors. ERRATOR is freely available from `https://perso.limsi.fr/wisniews/errator/`.
ERRATOR is made of two parts:

- a Python script that implements the method described in Section 2. to extract potential errors from one or several corpora in the CoNLL-U format. This implementation relies on our in-house Generalized Suffix Tree library that can represent an annotated corpus of sentences and efficiently handle all the required operations on strings.
- a web server that can be used to visualize and interacts with these potential errors. This server is developed using the micro web framework `Flask`[4] and relies on both Python and Javascript to generate the web pages and manage interactions with the user.

Figure 2 shows an example of the web interface. This interface allows the user to browse, sort, query potential annotation errors, highlighting the differences in annotations and link them to corresponding sentences in the UD corpora. It is also possible to filter out annotation variations that correspond to truly ambiguous sequence of words either by selecting a particular entry or by writing rules (regular expression that can match either the raw text, the annotations or the two). Annotations can not be corrected directly within ERRATOR so that users can use their favorite 'external' tools for annotating treebanks.

repeat length (words)	# repeats
2	71,349
3	65,764
4	31,274
5	12,053
≤ 5	206,296
≤ 10	+7,058
≤ 20	+359
≤ 84	+83

Table 2: Distribution of the match length in the French treebanks of the UD project.

4. Analyzing Annotation Errors in the French Corpora

In this section, we will present the results of our first experiments with ERRATOR to correct the errors on the different French treebanks of the Universal Dependencies project.
There are 5 treebanks for French in the UD project, representing a total of 40,101 sentences and 1,099,571 tokens. Considering all corpora, 213,796 sequence of two words or more that appear in more than one sentence, the longest repeat containing 84 words, and 7,000 repeats being made of more than 10 words. Table 2 gives some statistics on match length.
Table 3 shows the number of matches for which the annotation is not the same and are therefore considered as a potential error. Examples of the output produced by ERRATOR for the PoS annotation of the French `Sequoia` treebank are given in Figure 4.
To evaluate the impact of annotation incoherences on prediction performance, we have manually checked all the

[2] Formally, for a sentence of n words, the parse tree can be represented by a list `l` of n pairs so that `l[i]` is the index of the head of the i-th word of the sentence and the label of the dependence

[3] Considering alternative representations (e.g. by dropping labels or considering PoS rather than words) is left for future work.

[4] `flask.pocoo.org`

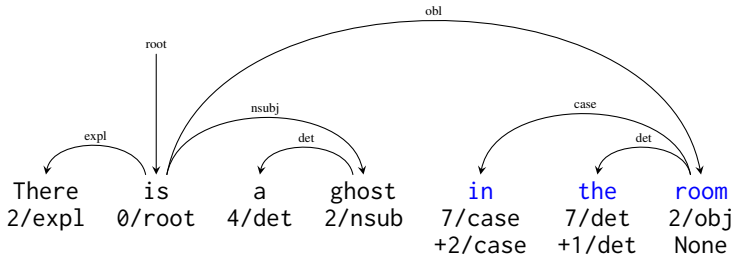

Figure 1: Representation of a dependency tree as a list of heads (first line) and representation of its substring 'on the issue' with local indices used when comparing to tree fragments (second line).

Figure 2: ERRATOR User Interface that allows users to browse, query and filter potential annotation errors.

annotation	number of potential errors
word segmentation	2,617
PoS	25,527
Dependencies	48,986

Table 3: Number of potential errors in the French treebanks of the UD project.

matches between the French UD and the French FTB correcting incoherences and errors.

The impact of these modifications on the performance of a PoS tagger are reported in Table 5. These result show, in particular, that, for PoS tagging, part of the drop in performance observed when testing on out-domain data is due to divergences in annotations, as shown by the comparison on the results in bold.

5. Conclusion

We have introduced ERRATOR, a set of tools implementing the *annotation variation principle* that can be used to help annotators find and correct errors in the different layers of annotations of UD treebanks. Our experience with ERRATOR to correct and harmonize the annotations of the different French corpora show that has proved the usefulness of this tool. ERRATOR could, however, be more efficient if it is tightly integrated with annotation tools to detect potential errors as soon as possible during the annotation campagn. This will be the goal of our future developments.

Acknowledgments

This work has been partly funded by the *Agence Nationale de la Recherche* (ParSiTi project, ANR-16-CE33-0021).

6. Bibliographical References

Aufrant, L., Wisniewski, G., and Yvon, F. (2017). Limsi@conll'17: Ud shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 163–173, Vancouver, Canada, August. Association for Computational Linguistics.

emea-fr-dev_00104	Pour les patients atteints d' un SCA , la dose initiale recommandée est un bolus intraveineux de 0,1 mg / kg suivi d' une perfusion de 0,25 mg / kg / h . ADP DET NOUN VERB ADP DET NOUN PUNCT DET NOUN ADJ VERB AUX DET NOUN ADJ ADP NUM NOUN ADP NOUN VERB ADP DET NOUN ADP NUM NOUN ADP NOUN ADP NOUN PUNCT
emea-fr-dev_00510	Pour les patients atteints d' un SCA , la dose initiale recommandée est : ADP DET NOUN VERB ADP DET NOUN PUNCT DET NOUN ADJ VERB VERB PUNCT
frwiki_50.1000_00767	- Michel Roussin , reconnu coupable de complicité et recel de corruption , a été condamné à 4 ans de prison avec sursis et une amende de 50 000 euros . PUNCT PROPN PROPN PUNCT AUX NOUN ADP NOUN CCONJ NOUN ADP NOUN PUNCT AUX AUX VERB ADP NUM NOUN ADP NOUN ADP NOUN CCONJ DET NOUN ADP DET NOUN PUNCT
frwiki_50.1000_00773	- Louise-Yvonne Casetta , trésorière occulte de le RPR , a été reconnue coupable de complicité et recel de corruption , et a été condamnée à 20 mois de prison avec sursis et 10 000 euros d' amende . PUNCT PROPN PROPN PUNCT NOUN ADJ ADP DET PROPN PUNCT AUX AUX AUX ADJ ADP NOUN CCONJ NOUN ADP NOUN PUNCT CCONJ AUX AUX VERB ADP NUM NOUN ADP NOUN ADP NOUN CCONJ DET NOUN ADP NOUN PUNCT
emea-fr-dev_00228	Les saignements majeurs se sont produits le plus fréquemment à le site de ponction (voir Tableau 3) . DET NOUN ADJ PRON AUX VERB DET ADV ADV ADP DET NOUN ADP NOUN PUNCT VERB PROPN NUM PUNCT PUNCT
emea-fr-dev_00258	Les saignements majeurs se sont produits le plus fréquemment à le site de ponction (voir Tableau 5) . DET NOUN ADJ PRON AUX VERB DET ADV ADV ADP DET NOUN ADP NOUN PUNCT VERB NOUN NUM PUNCT PUNCT

Table 4: Example of errors in PoS annotation found by ERRATOR on the French Sequoia corpus of the UD project. Words in blue and red appears in two different sentence, but the red part has not the same annotation.

Bosco, C., Montemagni, S., and Simi, M. (2013). Converting italian treebanks: Towards an italian stanford dependency treebank. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 61–69, Sofia, Bulgaria, August. Association for Computational Linguistics.

Boyd, A., Dickinson, M., and Meurers, W. D. (2008). On detecting errors in dependency treebanks. *Research on Language and Computation*, 6(2):113–137, Oct.

Dickinson, M. and Meurers, W. D. (2003). Detecting errors in part-of-speech annotation. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 107–114, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dickinson, M. and Meurers, W. D. (2005). Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 322–329, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Gusfield, D. (1997). *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

Lipenkova, J. and Souček, M. (2014). Converting russian dependency treebank to stanford typed dependencies representation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 143–147, Gothenburg, Sweden, April. Association for Computational Linguistics.

van Halteren, H. (2000). The detection of inconsistency in manually tagged text. In *Proceedings of the COLING-2000 Workshop on Linguistically Interpreted Corpora*, pages 48–55, Centre Universitaire, Luxembourg, August. International Committee on Computational Linguistics.

Vilares, D. and Gómez-Rodríguez, C. (2017). A non-projective greedy dependency parser with bidirectional lstms. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 152–162, Vancouver, Canada, August. Association for Computational Linguistics.

Wisniewski, G., Pécheux, N., Gahbiche-Braham, S., and Yvon, F. (2014). Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1779–1785, Doha, Qatar, October. Association for Computational Linguistics.

7. Language Resource References

Nivre, J., Agić, Ž., Ahrenberg, L., et al. (2017). Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague.

train	test	% error
FTB corrected	FTB corrected	2.96%
	FTB original	4.06%
	UD corrected	6.49%
	UD original	6.70%
UD original	UD original	4.51%
	UD corrected	4.53%
	FTB corrected	5.93%
	FTB original	**6.78%**
FTB original	FTB original	3.17%
	FTB corrected	3.88%
	UD corrected	6.91%
	UD original	7.01%
UD corrected	UD corrected	4.42%
	UD original	4.60%
	FTB corrected	**5.71%**
	FTB original	6.75%

Table 5: Comparison of the performance of a state-of-the-art PoS tagger on two different French dataset before and after the errors detected by ERRATOR have been corrected.