Corpus Correction

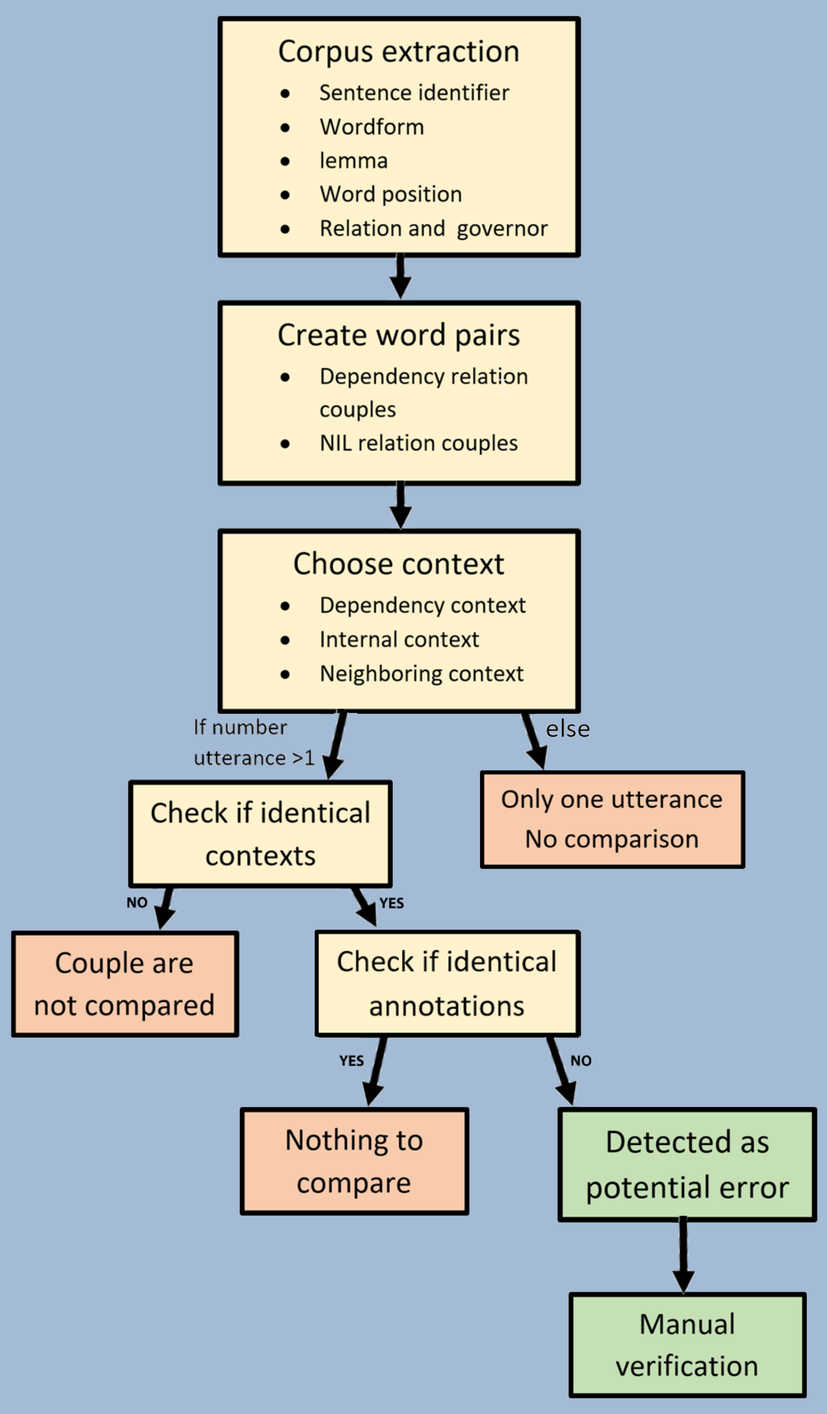
Margot GUETTIER, Kahina MENZOU, Papa Amadou SY
Supervisor: Bruno GUILLAUME

Objectives

Universal Dependencies (UD) is a project that is developing cross-linguistically consistent treebank annotation for more than 100 languages, 200 treebanks and count more than 300 contributors. The goal is facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective. What would happen if there were errors in the annotations?
It is in order to prevent this from happening that we have worked on this corpus correction project. This project consists in detecting and highlighting the potential annotation errors present in treebanks. To do this, we propose a semi-automatic method. The first part consists in detecting and extracting all the potential errors and the second part consists in an expert review to judge if they are really errors or not.

Methodology

The method for detecting error in corpus annotated with dependency relation is based on the principle of variation detection, that means if the same element is annotated twice differently we can assume that one of them might be an error. For this project the variation detection is done with regard to a chosen context and a couple of word.

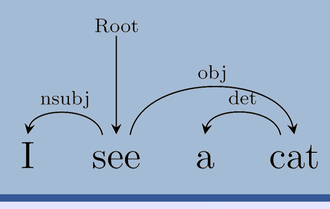


Corpus

The experimentation was carried out on the UD French GSD. We chosed the last version which is the 2.7, it is noted that all files are coded in Conllu. We used the **Train** file: **14449 sentences** and **354662 tokens**

Couple of words

Couples of word are of two kinds. Either with a dependency relation, such that one word is the dependant of the other word (e.g. *I* is the subject of *see*). Or with no relation, in this case we will call this a couple with NIL relation (e.g. *I* and *cat*)



References

[1] Adriane Boyd, Markus Dickinson, and Detmar Meurers. On detecting errors in dependency treebanks. *Research on Language and Computation*, 6:113–137, 10 2008.

Contexts

- Internal Context : all the elements between two words

Figure 1: Internal context

- Neighboring context : immediate context of both words

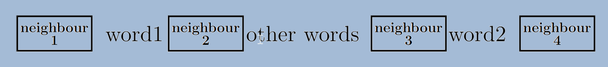


Figure: Neighboring context

- Dependency context : relation name between the governor and its own governor

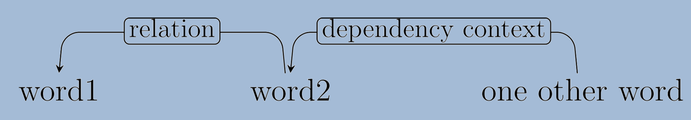


Figure 3: Dependency context

Results

The table below shows the results obtained for the different tests applied to the file in presence of wordform/lemma, NIL relation and punctuation.

	no context		Internal		Neighboring		Dependency	
	Wordform	Lemma	Wordform	Lemma	Wordform	Lemma	Wordform	Lemma
NIL/Punct	70549	78267	4076	4553	1070	1392	1050	2070
NIL/ not_Punct	55894	65508	2594	2836	910	1140	1050	2070
not_NIL/not_Punct	2704	4430	803	902	105	122	1050	2070

Table 1: Number of potential errors for each context

the figure below shows the representation of the display where errors are highlighted in red.

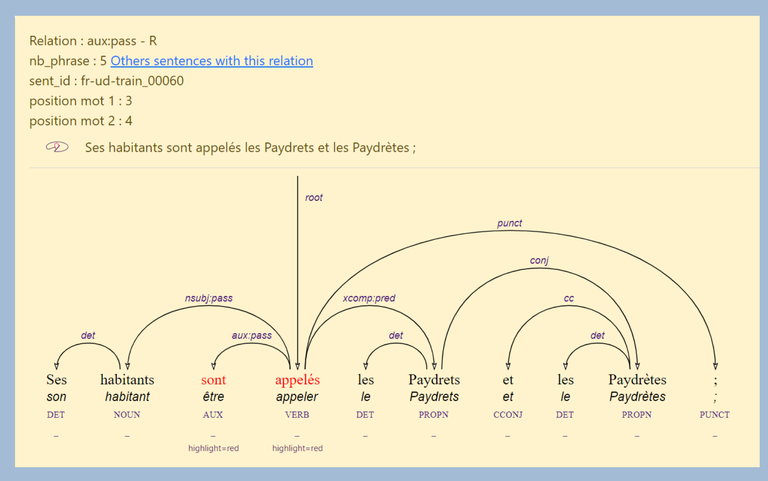


Figure 1: Figure caption

The conclusion of the manual evaluation is that the neighboring context is more interesting if we apply NIL: 50% of real errors are detected than the internal context with 30%. If we don't apply NIL the neighboring context is also the most interesting context with 75% of real errors followed by the internal context (65%) and Finally, the dependency context with 35%.

Conclusion

- The system actually manages to detect three different contexts of dependency errors (Neighboring, Internal, dependency) and with or without NIL relation but each one must be verify. The system is available and usable on any corpus coded in Conllu.

For future work, our goals are:

- Correct directly from the tool's display and apply modification operated on the HTML page on the conllu file. Another goal is to optimize the system so that it can takes less time when we have a large corpus.