

Auxiliary Tasks to Boost Biaffine Semantic Dependency Parsing

Marie Candito

LLF, Université Paris Cité - CNRS, Paris, France
marie.candito@u-paris.fr

Abstract

The biaffine parser of Dozat and Manning (2017) was successfully extended to semantic dependency parsing (SDP) (Dozat and Manning, 2018). Its performance on graphs is surprisingly high given that, without the constraint of producing a tree, all arcs for a given sentence are predicted independently from each other (modulo a shared representation of tokens). To circumvent such an independence of decision, while retaining the $O(n^2)$ complexity and highly parallelizable architecture, we propose to use simple auxiliary tasks that introduce some form of interdependence between arcs. Experiments on the three English acyclic datasets of SemEval 2015 task 18 (Oepen et al., 2015), and on French deep syntactic cyclic graphs (Ribeyre et al., 2014) show modest but systematic performance gains on a near state-of-the-art baseline using transformer-based contextualized representations. This provides a simple and robust method to boost SDP performance.

1 Introduction and related work

Semantic dependency parsing is the task of producing a dependency graph for a sentence. Depending on the datasets, these dependencies may correspond to predicate-argument relations, with labels numbering semantic arguments (as in Figure 1-top) or dependencies with intermediate status between syntax and semantics, with labels being canonical grammatical functions that normalize syntactic alternations (e.g. in Figure 1-bottom, the clitic *l'* (*him*) is the canonical object of the passive verb form *sollicité* (*solicited*)).

If one views each dependency as a decision to make, both dependency parsing (DP, outputting a syntactic tree) and semantic dependency parsing (SDP) are known to exhibit high interdependence of decisions. For instance in DP, when parsing *a question answering machine*, choosing *machine* as root is linguistically coherent with the *machine*

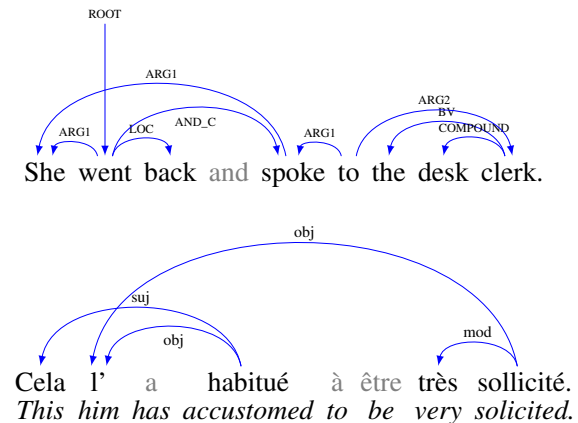


Figure 1: **Top:** English Semantic graph in the DM format, as part of the SemEval2015-Task18 dataset (Oepen et al., 2015). **Bottom:** French Deep syntactic graph as defined by Candito et al. (2014).

→ *answering* → *question* analysis only, whereas (wrongly) choosing *question* as root is syntactically coherent with the *question* → *answering* → *machine* analysis.

In DP though, the interdependence between arcs is partially solved by the tree constraint: choosing one head for a given token amounts to ruling out all other heads. This structural interdependence is absent in SDP. Complex structural, lexical and semantic factors control whether a given dependent should be attached to zero, one or several heads.

Several approaches exist in the literature to capture interdependence of arcs in SDP, which often derive from proposals made for DP. One is to use a higher-order graph-based parser. Wang et al. (2019) achieve state-of-the-art results (without pretrained LM) on the English part of the SemEval2015-Task18 data, by using second-order factors to score the graphs, yet at the cost of a $O(n^3)$ complexity.

Another main approach is to use sequential decisions, and hence take advantage of previous decisions by encoding the previously predicted arcs. This is the case in the transition-

based parser of Fernández-González and Gómez-Rodríguez (2020), or in the system of Kurita and Søgaard (2019), which selects a new head for certain tokens at each iteration, using reinforcement learning to order this selection of heads. Both models have a $O(n^2)$ complexity (when used without cycle detection), and in both cases, sequential decisions benefit from the encoding of previously predicted arcs, yet at the cost of error propagation. For that reason, Bernard (2021) propose a system close to that of Kurita and Søgaard (2019), yet allowing the system to overwrite previous decisions and hopefully correct itself.

On the contrary, the biaffine system of Dozat and Manning (2018) (hereafter **DM18**) performs a simultaneous scoring of all candidate arcs, decides to predict an arc independently of the other ones. This results in a highly parallelizable $O(n^2)$ inference, with surprisingly high performance albeit below second-order parsing.

As for most NLP tasks, SDP performance increases when integrating transformer-based contextual representations when encoding input tokens. On the English dataset from the SemEval 2015 Task 18 (Oepen et al., 2015), Fernández-González and Gómez-Rodríguez (2020) (hereafter **FG20**) report a +0.7 and +2.0 increase for the in-domain (ID) and out-of-domain (OOD) test sets respectively.¹

In this work, we retain the simple $O(n^2)$ biaffine architecture of DM18, and we investigate how simple auxiliary tasks can introduce some interdependence between arc decisions, in a multi-task learning setting (Caruana, 1997). We show modest but statistically significant improvements on the three English datasets of the widely used SemEval2015-Task18 data (Oepen et al., 2015). We also test another appealing property of the biaffine architecture, which is the absence of formal constraints on the output graphs. Experiments on French deep syntactic graphs (Ribeyre et al., 2014), which are highly cyclic, also demonstrate the effectiveness of our auxiliary tasks for SDP.

¹Using the biaffine DM18 architecture, He and Choi (2020) report a +2 and +3 point increase in ID and OOD. Yet, these results are not comparable: the authors have used a different pre-processing, which adds orphan dependencies from the root to orphan tokens, resulting in an easier task (p.c. with the authors and <https://github.com/emorynlp/bert-2019/issues/1>).

2 The baseline biaffine graph parser

We reuse the computation of the arc and label scores of the DM18 model, which we modernized by using contextual representations: input sequence $w_{1:n}$ is passed into a pretrained language model. We represent a word-token w_i by concatenating the contextual vector of its first subword² $\mathbf{h}_i^{(\text{bert})}$ and a word embedding $\mathbf{e}_i^{(\text{word})}$.

$$\mathbf{v}_i = \mathbf{h}_i^{(\text{bert})} \oplus \mathbf{e}_i^{(\text{word})} \quad (1)$$

For some of the experiments, we also concatenate a lemma and a POS embedding.

$$\mathbf{v}_i = \mathbf{h}_i^{(\text{bert})} \oplus \mathbf{e}_i^{(\text{word})} \oplus \mathbf{e}_i^{(\text{lemma})} \oplus \mathbf{e}_i^{(\text{POS})} \quad (2)$$

The sequence of word-tokens representations is passed into several biLSTM layers: $\mathbf{r}_{1:n} = \text{biLSTM}(\mathbf{v}_{1:n})$.

The recurrent representation r_i is then specialized according to two binary features: head versus dependent, and arc versus label score:

$$\begin{aligned} \mathbf{h}_i^{(\text{arc-head})} &= \text{MLP}^{(\text{arc-head})}(\mathbf{r}_i) \\ \mathbf{h}_i^{(\text{lab-head})} &= \text{MLP}^{(\text{lab-head})}(\mathbf{r}_i) \\ \mathbf{h}_i^{(\text{arc-dep})} &= \text{MLP}^{(\text{arc-dep})}(\mathbf{r}_i) \\ \mathbf{h}_i^{(\text{lab-dep})} &= \text{MLP}^{(\text{lab-dep})}(\mathbf{r}_i) \end{aligned} \quad (3)$$

We use a simplified biaffine transformation for arc scores, and a per-label one for label scores:

$$\begin{aligned} s_{i \rightarrow j}^{(\text{arc})} &= \mathbf{h}_j^{(\text{arc-dep})} \mathbf{U}^{(\text{arc})} \mathbf{h}_i^{(\text{arc-head})\top} + \mathbf{b}^{(\text{arc})} \\ s_{i \rightarrow j}^{(l)} &= \mathbf{h}_j^{(\text{lab-dep})} \mathbf{U}^{(l)} \mathbf{h}_i^{(\text{lab-head})\top} + \mathbf{b}^{(l)} \end{aligned} \quad (4)$$

For each position pair i, j , a binary cross-entropy loss is used for the existence of arc $i \rightarrow j$, and a cross-entropy loss is used for the labels of gold arcs. At inference time, any candidate arc with positive score $s_{i \rightarrow j}^{(\text{arc})}$ is predicted, and receives the label with maximum score for this arc.

3 Auxiliary tasks targeting sets of arcs

Preliminary experiments on English semantic dependency graphs (Oepen et al., 2015) and on French deep syntactic graphs (Candito et al., 2014) have shown us that the biaffine graph parser gives good results, but with inconsistencies that are clearly related to the locality of decisions. In particular, a quick error analysis revealed incompatible

²He and Choi (2020) report very slight improvements in OOD when using the average of all subwords, but an opposite trend in ID.

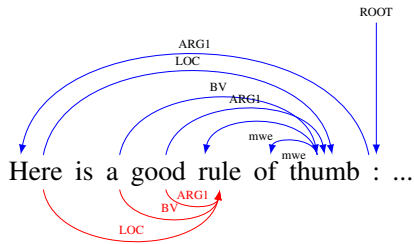


Figure 2: Example of competition for the sequence *rule of thumb*. Above arcs: correct MWE analysis (*rule* and *of* attached to the last MWE component *thumb*, and *thumb* being the head of the sequence). Below arcs: incorrect compositional analysis, in which *rule* is the head, e.g. attached wrongly as ARG1 of *good* (in red).

arc combinations. More precisely, we noticed impossible sets of labels for the set of heads of a given dependent. For example in the DM part of the SemEval2015-Task18 dataset, tokens are sometimes attached with a *mwe* label (for a component of a multi-word expression) and attached to another head with a non-mwe label, as shown for the *rule* token in Figure 2. This incorrect situation actually never happens in the training set, but this impossibility is not captured by the model. In the predicted French deep syntactic graphs, we noticed punctuation tokens wrongly attached to two different heads with the specific *punct* dependency label.

A second observation is a tendency in some of the datasets to predict disconnected tokens (i.e. with no incoming nor outgoing arcs) too frequently. More generally, when counting the number of predicted heads for each token in the predicted graphs, the accuracy is about 95% in the English datasets, and below 92% for the French one.

3.1 Auxiliary tasks

Hence the idea of using auxiliary tasks taking into account all the heads (resp. dependents) of a given token. More precisely, we experiment multi-task learning on the two target tasks (tasks **A** and **L** for arc and label prediction), plus the following auxiliary tasks, which predict for each token w_j :

- tasks **H** and **D**: the number of governors and number of dependents. For instance in top Figure 1, *spoke* has two governors (*went* and *to*) and one dependent (*She*);
- the labels of the incoming arcs, either as:
 - task **S**: the concatenated string of the incoming arcs labels, in alpha-

betic order (e.g. for the *spoke* token, AND_C+ARG1)

- task **B**: or the "bag of labels" (BOL) sparse vector, whose components are the numbers of incoming arcs to w_j bearing each label. For the *spoke* token, this would give a 1 for the AND_C label component and 1 for the ARG1 label, and zeros for all other labels.

Technically, for each auxiliary task, a specific MLP is used to specialize the recurrent representation r_j of each word-token w_j . Tasks H and D are regression tasks, which use MLPs with a single output neuron and a squared error loss.³

$$\begin{aligned} nbh_j &= \text{MLP}^{(H)}(\mathbf{r}_j) \\ nbd_j &= \text{MLP}^{(D)}(\mathbf{r}_j) \end{aligned}$$

The S task is a classification task into categories corresponding to multi-sets of labels encountered in the training set.⁴ For w_j , the vector of scores of all the label multi-sets is $\mathbf{s}_j = \text{MLP}^{(S)}(\mathbf{r}_j)$, and cross-entropy loss is used at training.

For the B task, we use a MLP with final layer of size $|L|$: $\mathbf{BOL}_j = \text{MLP}^{(B)}(\mathbf{r}_j)$. The component for label l , BOL_{jl} , is interpreted as $1 + \log$ of the number of l -labeled incoming arcs to w_j . The loss we use is the L2 distance between gold and predicted BOL vectors.

The two example inconsistencies cited above are indirectly captured by these auxiliary tasks. Firstly, in case of a token w_j that is a component of a multi-word expression, the recurrent representation \mathbf{r}_j for this token will be optimized to lead to a single *mwe* incoming label for the S or B tasks, and a value 1 for the H task (cf. a single governor for w_j). Hopefully, when used for the A and L tasks, \mathbf{r}_j will favor incoming arcs from close next tokens (cf. e.g. for the DM format, *mwe* components are attached to the right, and quite locally). The other mentioned problem of predicting disconnected tokens too frequently is captured by the H and D tasks. Predicting no incoming nor outgoing arc for a given token will only be coherent with H=0 and D=0 for this token. Hence, hopefully, predicting

³ nbh_j is interpreted as $1 + \log$ of the number of heads of w_j , and same for nbd_j , so as to penalize less errors in bigger numbers: e.g. penalize more predicting one head instead of 0 than predicting 2 heads instead of 3.

⁴The categories are label multi-sets because we neutralize the order of the heads when considering the incoming arcs. This limits the number of categories.

more than 0 for the H task for a token w_j , will lead to higher scores for arcs pointing to w_j .

3.2 Combining sublosses

At training time, for each batch, we seek to minimize a weighted sum of the losses for all the tasks, whether main or auxiliary. Manually tuning these weights is cumbersome and suboptimal. We use the notion of task uncertainty and the approximation proposed by Kendall et al. (2018), who introduce a parameter σ_t for each subtask t , to be interpreted as its ‘‘uncertainty’’. Noting T as the set of tasks, the overall loss for a batch is $\sum_{t \in T} \frac{1}{\sigma_t^2} L_t + \ln(\sigma_t)$.

The parameters σ_t are initialized to 1 and modified during the learning process. The first term of the sum ensures that the more uncertain the task, the less its loss will count, while the second term prevents arbitrarily augmenting the σ_t values, thus reducing the loss weights.

3.3 Stack propagation

We test two multi-task learning configurations: first, simple parameter sharing up to the biLSTM layers (all specialization MLPs applied on the recurrent token representations). Second, we test the technique of ‘‘stack propagation’’, which Zhang and Weiss (2016) experimented for the POS tagging and parsing tasks. In our case, it amounts to using the dense layers of the auxiliary tasks MLPs to score the arcs and their labels.

For example, to use the H task in stack propagation mode, let $\text{hidden}_j^{(H)}$ be the hidden layer of $\text{MLP}^{(H)}$ for the dependent j , and $c^{(H)}$ a coefficient hyperparameter. The computation of $s_{i \rightarrow j}^{(\text{arc})}$ (cf. equation 4) is modified as follows:

$$\begin{aligned} \text{sp}_j^{(\text{arc-dep})} &= \mathbf{h}_j^{(\text{arc-dep})} \oplus c^{(H)} \text{hidden}_j^{(H)} \\ s_{i \rightarrow j}^{(\text{arc})} &= \text{sp}_j^{(\text{arc-dep})} \mathbf{U}^{(\text{arc})} \mathbf{h}_i^{(\text{arc-head})\top} + \mathbf{b}^{(\text{arc})} \end{aligned}$$

Similarly, to use task B in stack propagation mode, we modify the score of each label:

$$\begin{aligned} \text{sp}_j^{(\text{lab-dep})} &= \mathbf{h}_j^{(\text{lab-dep})} \oplus c^{(B)} \text{hidden}_j^{(B)} \\ s_{i \rightarrow j}^{(l)} &= \text{sp}_j^{(\text{lab-dep})} \mathbf{U}^{(l)} \mathbf{h}_i^{(\text{lab-head})\top} + \mathbf{b}^{(l)} \end{aligned}$$

Note that it forces to perform the auxiliary tasks during inference, instead of at training time only.

4 Experiments and discussion

4.1 Datasets

We experiment on the three widely used English datasets of SemEval2015-Task18 (Oepen et al., 2015) (DM, PAS and PSD), which are acyclic

graphs mainly representing predicate-argument relations. We also experiment on French deep syntactic graphs (Ribeyre et al., 2014) (Appendix D). These capture most of argument sharings (e.g. raising, obligatory and arbitrary control, subject sharing in VP coordination) but are closer to surface syntax in the sense that labels remain syntactic, even though syntactic alternations are neutralized (e.g. passive *by*-phrases are labeled as subjects). Cycles may appear e.g. in relative clauses.⁵

4.2 Experimental protocol

We chose to investigate the impact of the auxiliary tasks on a high baseline, using pretrained contextual representations. We use our own implementation⁶ of the biaffine parser, the $\text{BERT}_{\text{base-uncased}}$ model for English, and $\text{FlauBERT}_{\text{base-cased}}$ for French.⁷ We used two settings (see Appendix A for details):

- **BERT_{tuned}**: the first setting is intended to use the contextual representations as only source of pre-trained parameters, and defines the \mathbf{v}_i vectors as in equation (1) (no lemma nor POS embeddings), with the word embeddings being randomly initialized, and the BERT embeddings being fine-tuned for the SDP task.
- **BERT_{froz}+POS+lem**: the second one is used to compare our results to previous work on the English SemEval2015-Task18 datasets: the BERT embeddings are frozen, additional POS and lemma embeddings are used (cf. \mathbf{v}_i definition as in equation (2)). The same pre-trained word and lemma embeddings as FG20 are used. Note this setting uses gold POS and lemmas and is not a realistic scenario.

For the **BERT_{tuned}** mode, we tuned the hyperparameters on the French data, and applied the same configuration to the English datasets. After a few tests, we set a configuration (see Appendix A), and searched for the best combination of auxiliary tasks. For each experiment, we report the labeled Fscore (LF), including root arcs, averaged over 9 runs.

⁵In these deep graphs, the root tokens (usually unique) are attached to a dummy root token in practise. Thus for this dataset, in all the above formulations, the sequence $w_{1:n}$ corresponds to a sentence of $n - 1$ word-tokens, with a dummy root w_1 . For its contextual representation, we use the contextual vector of the beginning of sequence token.

⁶<https://github.com/mcandito/aux-task-s-biaffine-graph-parser-FindingsACL22>

⁷We used the HuggingFace library (Wolf et al., 2020).

4.3 Results on French deep syntactic graphs

Table 1 shows the results with and without various combinations of auxiliary tasks.⁸ While no auxiliary task provide a significative increase⁹, the combinations B+H and B+D+H+S bring a statistically significant +0.53 and +0.56 increase on average (see Appendix C for significance test).

Auxiliary tasks	Stack propagation	On 9 runs	
		meanLF	stdev
\emptyset	NA	86.79	0.19
B+H	no	87.32***	0.18
D+H	no	86.61	0.71
H+S	no	87.04	0.17
B+D+H+S	no	87.35***	0.26
B+H	$c^{(B)}=1$ $c^{(H)}=1$	87.49	0.06
B+H	$c^{(B)}=1$ $c^{(H)}=10$	87.66+++	0.18

Table 1: Results on French dev set, for various tasks combinations, with and without stack propagation (H: nb of heads, D: nb of dependents, B: bag of labels, S: label multi-set). Col3-4: average LF, and standard deviation. ***: significative diff. wrt first line. +++: significative diff. wrt second line ($p < 0.001$).

We tested the impact of stack propagation using auxiliary tasks B+H. We observe a modest but statistically significant +.34 increment with weights $c^{(B)}=1$ and $c^{(H)}=10$. Considering that this makes the inference task more complex, we did not use it in later experiments on English.

4.4 Results on English semantic graphs

	Tasks	DM	PAS	PSD	Avg
ID	\emptyset	93.7	93.9	80.7	89.4
	B+H	94.2	94.3	81.2	89.9
OOD	\emptyset	90.3	92.0	79.8	87.4
	B+H	91.0	92.8	80.2	88.0

Table 2: Average LF (on 9 runs), in BERT_{tuned} setting, on English in-domain (ID) and out-of-domain (OOD) test sets, using either no auxiliary task (\emptyset) or tasks B and H (B+H), without stack propagation. B+H results are statistically higher than \emptyset for DM ID, DM OOD, PAS ID, PAS OOD ($p < 0.001$) and PSD ID ($p < 0.01$).

⁸Previous state-of-the art on this data is a non-neural system: Ribeyre et al. (2016) obtained LF=80.86, and went up to LF=84.91 thanks to features from constituency parses from the rich FrMG parser (Villemonte De La Clergerie, 2010). The biaffine architecture with contextual vectors, without auxiliary tasks, obtains a mean LF=86.79.

⁹See Appendix D for results with each auxiliary task.

We then tested the B+H configuration on the English test sets. In Table 2, we observe that performance gains using B and H auxiliary tasks are systematic across datasets (DM, PAS, PSD) and across in- or out-of-domain test sets, which tends to show the robustness of our method.¹⁰

We can also measure the impact of the auxiliary tasks by evaluating how accurate the predicted graphs are, concerning the number of heads of tokens: on average on the English dev sets, the proportion of tokens receiving the right number of heads in the predicted graphs increases from 94.9 without auxiliary tasks to 95.5 with tasks B+H.

Finally, we provide in Table 3 a comparison to FG20 results (thus using the BERT_{froz}+POS+lem setting), which are the state-of-the-art for systems using a single source of contextual embeddings. While our results remain below, note that our auxiliary tasks can be used with their system, as well as with e.g. that of Wang et al. (2021), which achieve significant improvements with an automated concatenation of various contextual embedding models, reaching 91.7 for ID et 90.2 for OOD.

	ID	OOD
FG20 BERT _{froz} +POS+lem	90.7	88.8
Ours BERT _{froz} +POS+lem, B+H	90.2	87.9
Ours BERT _{tuned} , B+H	89.9	88.0

Table 3: Comparison to the state-of-the-art SDP parser using BERT, on English ID and OOD test sets, in BERT_{froz}+POS+lem setting. **FG20**: (Fernández-González and Gómez-Rodríguez, 2020).

5 Conclusion

When using a biaffine graph-based architecture for semantic dependency parsing (SDP), arcs are predicted independently from each other. Our contribution is a set of simple yet original auxiliary tasks that introduce some form of interdependence of arc decisions. We showed that training recurrent word-token representations both for the SDP task and for predicting the number of heads and the incoming labels of each word is systematically beneficial, when tested either on English or French, on semantic or on deep syntactic graphs, and on in- or out-of-domain data.

¹⁰The improvement tends to be higher for OOD, and for DM and PAS. One reason could be that the PSD graphs show less reentrancies, hence the number of heads is more predictable, and using a specific auxiliary task for it is less beneficial.

References

- Anne Abeillé and Nicolas Barrier. 2004. [Enriching a French treebank](#). In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).
- Timothée Bernard. 2021. [Multiple tasks integration: Tagging, syntactic and semantic parsing as a single task](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 783–794. Online. Association for Computational Linguistics.
- Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie. 2014. [Deep syntax annotation of the sequoia French treebank](#). In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2298–2305, Reykjavik, Iceland. European Language Resources Association (ELRA).
- Rich Caruana. 1997. [Multitask learning](#). *Machine learning*, 28:41–75.
- Timothy Dozat and Christopher D. Manning. 2017. [Deep Biaffine Attention for Neural Dependency Parsing](#). In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France. OpenReview.net.
- Timothy Dozat and Christopher D. Manning. 2018. [Simpler but more accurate semantic dependency parsing](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.
- Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. [Transition-based semantic dependency parsing with pointer networks](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7035–7046. Online. Association for Computational Linguistics.
- Han He and Jinho Choi. 2020. [Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with BERT](#). In *Proceedings of Florida Artificial Intelligence Research Society Conference FLAIRS-33*, pages 228–233.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. [Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics](#). In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.
- Shuheì Kurita and Anders Søgaard. 2019. [Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy. Association for Computational Linguistics.
- Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. [Stack-pointer networks for dependency parsing](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414, Melbourne, Australia. Association for Computational Linguistics.
- Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. [SemEval 2015 task 18: Broad-coverage semantic dependency parsing](#). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.
- Corentin Ribeyre, Marie Candito, and Djamé Seddah. 2014. [Semi-Automatic Deep Syntactic Annotations of the French Treebank](#). In *Proceedings of the 13th International Workshop on Treebanks and Linguistic Theories (TLT13)*, pages 184–197, Tübingen, Germany.
- Corentin Ribeyre, Eric Villemonte de la Clergerie, and Djamé Seddah. 2016. [Accurate deep syntactic parsing of graphs: The case of French](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3563–3568, Portorož, Slovenia. European Language Resources Association (ELRA).
- Éric Villemonte De La Clergerie. 2010. [Convertir des dérivations TAG en dépendances](#). In *Actes de la 17e conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, pages 91–100, Montréal, Canada. ATALA.
- Xinyu Wang, Jinxian Huang, and Kewei Tu. 2019. [Second-order semantic dependency parsing with end-to-end neural networks](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.
- Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. [Automated concatenation of embeddings for structured prediction](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660. Online. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System*

Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Yuan Zhang and David Weiss. 2016. [Stack-propagation: Improved representation learning for syntax](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

A Training details

All experiments are run on a Nvidia GTX 1080 Ti GPU.

For all settings:

- lexical embedding size ($e_i^{(word)}$): 100
- lexical dropout : 0.4. At learning, the lexical embedding of a token is replaced with probability 0.4 by a special token *DROP*, whose embedding is learned.
- biLSTM : 3 layers of size $2 * 600$, with 0.33 dropout
- MLPs for aux. tasks: 1 hidden layer (300), output layer (300), dropout 0.25
- Batch size = 8
- Optimizer = Adam, $\beta_1 = \beta_2 = 0.9$
- Learning is stopped when all labeled Fscores decrease on the dev set. Note that beside the main FL score, tasks H and B give rise to their labeled Fscore, computed using the number of heads as predicted by task H (resp. B).

BERT_{tuned} setting

- BERT and FlauBERT models : fine-tuned
- MLPs for arc and label score : 1 hidden layer (600), output layer (600), dropout 0.33
- Learning rate = 2×10^5
- Loss combination : learnt weights (cf. section 3.2)

BERT_{froz}+POS+lem setting

- BERT model : frozen
- word and lemma embeddings (100), initialized with embeddings by [Ma et al. \(2018\)](#), fine-tuned

- POS embeddings (100), randomly initialized
- MLP for arc score : 1 hidden layer (500), output layer (500), dropout 0.33
- MLP for label score : 1 hidden layer (100), output layer (100), dropout 0.33¹¹
- Learning rate = 5×10^4
- loss combination : plain sum of losses for each task

B Unsuccessful tests

Various tests were abandoned as unsuccessful in our preliminary tests:

- Using pre-trained lexical embeddings with tuned contextual embeddings had no impact on performance on average.
- Freezing BERT’s and FlauBERT’s parameters without using word and lemma embeddings significantly decreased performance (by about 2 FL points).
- Increasing the level of parameter sharing between tasks was not successful: instead of applying the MLPs of the auxiliary tasks on the recurrent representations \mathbf{r}_i , we tested applying them on the outputs of the specialization MLPs (i.e. on $\mathbf{h}_i^{(\text{arc-head})}$ for task D, on $\mathbf{h}_i^{(\text{arc-dep})}$ for task H, on $\mathbf{h}_i^{(\text{lab-dep})}$ for tasks B and S). While this tends to increase the number of epochs, it does not improve the performance.

C Significance testing

We use a Fisher-Pitman exact permutation test to estimate the significance of the differences in performance between two configurations (as done for example by [Bernard, 2021](#)). More precisely, suppose we consider two samples of Fscores, for nA runs corresponding to configuration A, and nB runs for configuration B, with on average configuration B better than A. The null hypothesis is that the two samples follow the same distribution. The p-value corresponds to the probability that separating the set of Fscores into two samples A’ and B’ of size nA and nB gives a difference in mean at

¹¹Sizes of the MLPs for arc and label scores are defined here to replicate [\(He and Choi, 2020\)](#) settings. We observed very marginal differences when keeping the sizes used in BERT_{tuned} setting (600 for both MLPs).

least as large as the observed difference. With the exact test, the p-value is calculated exactly, on all possible splits into A' and B' samples.

D French data statistics and full results

	Train	Dev
Nb of sentences	14,759	1,235
Nb of tokens	457,872	40,055
% of disconnected tokens	12.0	12.2
Nb of edges	424,813	37,110

Table 4: Statistics of the deep French syntactic graphs, built on the French treebank (Abeillé and Barrier, 2004).

The complete results on dev set for the French data is provided in Table 5, of which Table 1 is a truncated version.

Auxiliary tasks	Stack propagation	On 9 runs	
		meanLF	stdev
∅	NA	86.79	0.19
H	no	86.82	0.54
D	no	86.83	0.40
S	no	86.98	0.30
B	no	87.05	0.49
B+H	no	87.32***	0.18
D+H	no	86.61	0.71
H+S	no	87.04	0.17
B+D+H+S	no	87.35***	0.26
B+H	$c^{(B)}=1$ $c^{(H)}=1$	87.49	0.06
B+H	$c^{(B)}=1$ $c^{(H)}=10$	87.66+++	0.18

Table 5: Full results on French dev set, for various tasks combinations, with and without stack propagation (H: nb of heads, D: nb of dependents, B: bag of labels, S: label multi-set). Col3-4: average LF, and standard deviation. ***: significant difference wrt first line ($p < 0.001$). +++: significant difference wrt B+H without stack propagation ($p < 0.001$).