Unifying parsing and tree-structured models for generating sentence semantic representations

Anonymous EACL submission

Abstract

The paper introduces a novel tree-based model that learns its composition function together with its structure. The proposed architecture produces sentence embeddings by composing words according to an induced syntactic tree. The parsing and the composition functions are explicitly connected and, therefore, learned jointly. As a result, the sentence embedding is computed according to an interpretable linguistic pattern and may be used on any downstream task. Since the composition model does not require an external parser, we explore to which extent we can take advantage of this property to reduce the need for external syntactic annotations. Finally we evaluate our encoder on a textual entailment task and we observe that it outperforms tree-based models relying on external parsers and is even competitive with BERT-base model trained on datasets many orders of magnitude larger.

1 Introduction

Computing sentence semantic representations traditionally calls for a recursive compositional function whose structure is tree-shaped. However, alternative methods have been used for computing semantic representations, such as recurrent neural network (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) or BERT-based representations (Devlin et al., 2019). These methods have gained increased popularity within the community because, contrary to tree-based models, they do not need carefully hand-annotated data to be trained. On the other hand, as many results suggest (Linzen et al., 2016; Jawahar et al., 2019; Clark et al., 2019), these new models acquire some sort of tree structure.

Rather than analyzing the patterns learned by those statistical models, we propose a framework relying explicitly on syntactic trees. To limit the need for an external parser, we propose a unified architecture, which infers an explicit tree structure and trains recursively a sentence embedding model.

We study the relevance of this framework for semantic inference. In particular we aim to ***(i)*** quantify the properties of such tree-structured models for downstream tasks and ***(ii)*** compare the learned structures with interpretable annotations. We use a textual entailment task to analyze how the structure can be used without task-specific tree annotations. We show that it is competitive with BERT-base with a significantly smaller training set.

2 Model

We introduce a model that performs jointly sentence parsing and the prediction of a sentence embedding. The sentence embedding is predicted by a weighted TREE-LSTM whose tree structure is provided by a dependency parser. The TREE-LSTM recursive composition function crucially uses a weighted sum of the child representations whose weights are provided by the parser edges, hence linking the parser outputs to the TREE-LSTM recursion. The model is illustrated in Figure 1.

Parsing model The parser is a standard graph based biaffine dependency parser (Dozat and Manning, 2017). It is formalized in two steps. First, in Eq. 2 to 4, it computes a weight matrix that is interpreted as weighted directed graph whose nodes are the sentence tokens:

$$\text{Biaff}(x_1, x_2) = x_1^T U x_2 + W^{(b)}(x_1 \oplus x_2) + b^{(b)} \quad (1)$$

$$a_k^{(dep)} = W^{(dep)} h_k + b^{(dep)} \quad (2)$$

$$a_j^{(head)} = W^{(head)} h_j + b^{(head)} \quad (3)$$

$$s_{kj}^{(arc)} = \text{Biaff}(a_k, a_j) \quad (4)$$

The second step performs parsing by computing a maximum spanning tree from the graph. As in Dozat and Manning (2017), we use the MST algo-

rithm to ensure the well-formedness of the tree:

$$\alpha_{kj} = \mathbb{1}_{mst(s_{kj}^{(arc)})} s_{kj}^{(arc)} \quad (5)$$

where α_{kj} is the probability of the edge linking node j to node k. For a given node k, there is at most one non-zero weighted edge leading to its governor j.

Compositionally weighted tree LSTM Given a predicted tree structure, we recursively encode the sentence using a variant of the Child Sum Tree model from (Tai et al., 2015) The recursive function is detailed below:

$$\tilde{h}_j = \sum_{k \in C(j)} \alpha_{kj} h_k, \quad (6)$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \quad (7)$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \quad (8)$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \quad (9)$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \quad (10)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \quad (11)$$

$$h_j = o_j \odot \tanh(c_j), \quad (12)$$

where in Eq. 6, $C(j)$ denote the set of children of node j and $k \in C(j)$. Crucially, in our case, Eq. 6 is a weighted sum rather than a standard sum and the weights used are those α_{kj} provided by the parser.

Discussion The core model outputs a sentence embedding computed by a weighted TREE-LSTM. The tree shape and the edge weights are given by the best prediction of a graph parser. The parsing model is linked to the TREE-LSTM by the weights α_{kj}. This architecture thus allows us to train jointly the parser and the TREE-LSTM.

The method described here differs from previous work as the representation is not computed from the whole forest of potential trees (Yogatama et al., 2017; Maillard et al., 2019; Williams et al., 2018; Shen et al., 2018) but instead on the single best tree while being completely differential. Thus, it can be trained using back-propagation on a supervised task for which we want to provide relevant sentence embeddings. The architecture is, in principle, generic and adaptable to any other downstream task.

3 Evaluation on downstream tasks

Our architecture primarily aims to produce relevant embeddings for downstream tasks. To assess our framework's efficiency, we evaluate it on the **SICK-R** downstream task (Marelli et al., 2014). This is a SentEval task (Conneau and Kiela, 2018) specifically dedicated to evaluate compositional semantic models. It consists of a textual entailment task made of sentence pairs annotated for semantic similarity on a 1 to 5 real range. The dataset consists of 9927 sentence pairs distributed in a 4500/500/4927 train/dev/test split. It includes specific examples of variations on passive and active forms, quantifier and modifier switches or negations. The task intends to discriminate models that rely on shallow lexical pattern matching from those that take advantage of the sentence underlying structure.

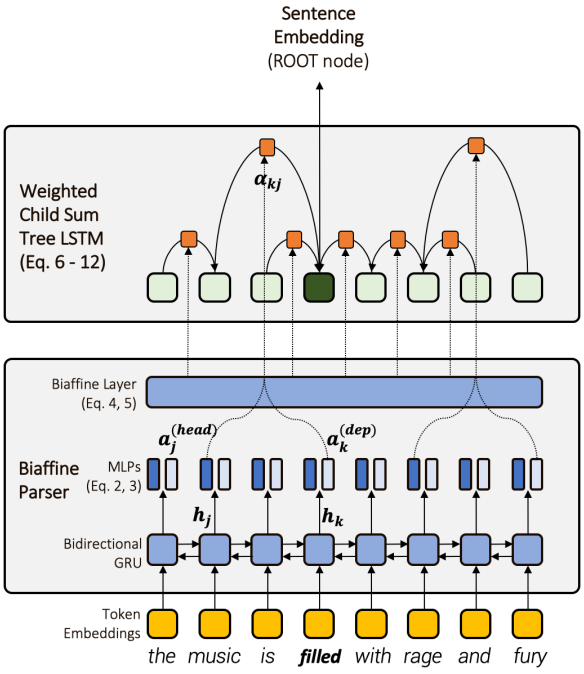

Figure 1: **Model architecture:** We illustrate Eq. 2 to 12. The Biaffine parser provides the structure of the sentence from which the TREE-LSTM computes sentence embeddings. The full pipeline is differentiable as the TREE-LSTM weights are given by the parser.

3.1 Training configuration

We use a similar training procedure[1] as in (Tai et al., 2015). We transform the target y from the **SICK-R** task into the distribution p defined by:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

[1] Hyperparameters such as hidden size, optimization procedure such or learning rate are fixed as proposed in Tai et al. (2015) and are detailed in Appendix.

A dedicated architecture is used to predict the similarity distribution from a pair of sentences. The similarity module takes as input a pair of sentence vectors h_L and h_L and computes their component-wise product $h_L \odot h_R$ and their absolute difference $|h_L - h_R|$. Given these features, we compute the probability distribution $\hat{p}_\theta$ using a two layers perceptron network (MLP):

$$\begin{aligned} h_\times &= h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\ h_s &= \sigma(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}), \\ \hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}), \end{aligned} \tag{13}$$

We use the KL-divergence between the prediction $\hat{p}_\theta$ and the ground truth p as training objective:

$$J(\theta) = \frac{1}{N} \sum_{k=1}^{N} \text{KL}(p^{(k)} \Big|\Big| \hat{p}_\theta^{(k)}) + \frac{\lambda}{2} ||\theta||_2^2 \tag{14}$$

Finally during inference, the similarity score $\hat{y}$ is computed as $\hat{y} = r^\intercal \hat{p}_\theta$ with $r^\intercal = [1, \ldots, 5]$.

3.2 Comparison with other encoders

We compare our architecture with other standard models from the literature. For this comparison, we first pre-train the parsing submodel on human-annotated sentences from the Penn Tree Bank (PTB) (Marcus et al., 1993) converted to Stanford dependencies. We then fine tune the parser's parameters on the **SICK-R** dataset while training the full model. We perform training for a maximum of 20 epochs and stop when no improvement was observed on the dev set for 3 consecutive epochs. We report the corresponding results from the test set in Table 1.

As expected, structured models perform better than models using weaker underlying structure.

We also observe that our model is competitive with a BERT-base upper-line. It is essential to note that BERT models are heavily pre-trained on vast corpora, whereas our structured models are trained only on the **SICK-R** and PTB data. Moreover, our model achieves a significantly lower standard variation than BERT-base implementation.

Encoder	r	ρ	MSE
BOW	78,2 (1,1)	67,9 (1.6)	40.0 (1.9)
LSTM	84.6 (0.4)	78.4 (0.5)	29.2 (0.8)
Bidirectional LSTM	85.1 (0.4)	78.7 (0.4)	28.5 (0.7)
N-ary TREE-LSTM	85.3 (0.7)	79.3 (0.6)	28.2 (1.4)
ChildSum TREE-LSTM	86.5 (0.4)	80.5 (0.5)	25.9 (0.8)
BERT-base	87.3 (0.9)	81.5 (1.4)	25.7 (2.3)
BERT-large	89.4 (0.6)	84.7 (1.0)	21.5 (1.3)
Our model			
Unified TREE-LSTM	87.0 (0.3)	81.3 (0.3)	24.9 (0.6)

Table 1: **Comparison with other models:** We compare our architecture with other frameworks on the **SICK-R** task. We report the average score on 5 runs (standard deviations in parentheses). Results are grouped as follow: **(1)** No underlying structure **(2)** Sequential structure **(3)** Tree structured models introduced in Tai et al. (2015) **(4)** BERT models from Devlin et al. (2019) **(4)** Our model with a pre-training on the PTB. We report Pearson and Spearman correlations from the test set, by convention as respectively r and $\rho \times 100$.

4 Experiments in low supervised setups

For languages such as English, linguistic resources are available, and it is technically possible to pre-train the parser. However, ressources such as the PTB are not available in all languages. That's why we investigate to which extent we can limit the syntactic supervision to infer the sentence's underlying structure. To this end, we contrast distinct configurations for which we initialize our parser with various degrees of supervision[2].

- In the **PTB-All** configuration, the parser is previously pre-trained on the PTB. This configuration is the same as in Section 3.2.
- In the **PTB-$\emptyset$** configuration, the parser parameters are randomly initialized[3].
- We also consider an initialization with only a small proportion of the **PTB** and train a parser by only using 100 randomly selected samples. This configuration is referred as **PTB-100**.

In all configurations, we either continue to update the parser when fine-tuning the model on downstream tasks or freeze the parser and only train the TREE-LSTM. These two configurations are indicated with respectively ✓ and × symbols.

[2] In this configuration, we observe pre-training the parser may cause weights α to become too large in equation 5. This leads to poor downstream performances. We correct this with a multiplicative parameter τ whose value is estimated during training. It means we replace Eq. 5 with: $\alpha_{kj} = \tau \cdot \mathbb{1}_{mst(s_{kj}^{(arc)})} s_{kj}^{(arc)}$ for tree weights computation.

[3] Parameters initialization procedure is detailed in Annexes.

4.1 Impact of the parser initialization

To quantify the effect of the parser configuration on downstream tasks, we compare the impact of the distinct setups on the **SICK-R** task. We report the Results on the **SICK-R** dev set in Table 2.

PTB sample size	Parser fine-tuning	Pearson Correlation (r)
PTB-$\emptyset$	✓	85.6 (0.6)
PTB-100	×	86.6 (0.2)
PTB-100	✓	86.9 (0.4)
PTB-All	×	87.2 (0.2)
PTB-All	✓	87.5 (0.4)

Table 2: **Impact of the parser initialization:** We train the parser module with a given sample size from the PTB. We either freeze (×) or update (✓) the parser during the fine-tuning on the **SICK-R** task. We report the average score on the dev set from 5 runs (standard deviations in parentheses). We report Pearson correlation by convention as $r \times 100$.

As expected, models using the full annotated data reach the top scores. However, using only a sub-sample significantly improves the results. Even in this low-supervised configuration, these models still outperform those using weaker underlying structures such as LSTM (Section 3.2). Indeed the corresponding Pearson correlation score for the LSTM on the dev set is 84.6 (1.3). This setup suggests that our architecture is particularly relevant for languages for which linguistic annotations are not massively available.

Moreover, fine-tuning the parser on the task leads to an improvement of the downstream results. We hypothesize that despite the modern parsers accuracy, the predicted tree is not guaranteed to be correct. Even if it is correct, it might not be the best fit from a task-oriented point of view. Fine-tuning the parser on the downstream tasks might enable the model to adapt itself to the task-specific dataset.

4.2 Analysis of the parses

In the first place, our framework aims to be a structured sentence encoder. However, we are also interested in interpreting the structures the model actually learns. To this end, we compare the parses generated by two distinct models differing by their initialization. In that regard, we compare the parses on the dev set of the **SICK-R** task. The results are presented in Table 3. We measure the Unlabeled Attachment Score (UAS) between the two parsers, that is the ratio from the number of common arcs between two parses by the total number of arcs.

Parser 1	Parser 2	UAS
Impact of the PTB sample size		
PTB-100 (✓)	PTB-$\emptyset$ (✓)	6,3
PTB-All (✓)	PTB-$\emptyset$ (✓)	10.1
PTB-All (✓)	PTB-100 (✓)	76.9
Impact of parser fine-tuning		
PTB-100 (✓)	PTB-100 (×)	85.2
PPTB-All (✓)	PTB-All (×)	98.4

Table 3: **Comparison of the parses:** We compare the parses from the **SICK-R** dev set, using distinct parser configurations. UAS scores correspond to the pairwise comparison from the corresponding configurations.

The UAS between PTB-$\emptyset$ and PTB-All are extremely low. This reveals that the parses obtained with only downstream task supervision have few in common with gold linguistic parses[4]. Interestingly, PTB-All and PTB-100 are remarkably close. This indicates that only a few PTB samples are required to obtain intelligible linguistic parses.

Regarding the PTB-100 configuration, we note an evolution of the parses when fine-tuning on the downstream task. As already observed in the previous experiment, we hypothesize the model can adapt itself to the dataset's specificity.

5 Conclusion and future work

We investigate the relevance of incorporating tree-like structural bias in the context of sentential semantic inference. To this end, we formulate an original model for learning tree-structure with distant downstream supervision. We evaluate our model on a textual entailment task and outperform sequential models and tree-structured models relying on external parsers. When initialized on human-annotated structure, it improves inference close to BERT performances.

In addition, we observe the sole use of downstream supervision is not sufficient to produce parses that are easy to interpret from a linguistic point of view. We believe the dataset is probably too small to allow the generalisation to more relevant structures. However, we observe that a limited amount of syntactic supervision is sufficient to get readable structures.

[4] in Annexes, we present the obtained trees for some examples.

References

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of bert's attention. *CoRR*, abs/1906.04341.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3651–3657.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL*, 4:521–535.

Jean Maillard, Stephen Clark, and Dani Yogatama. 2019. Jointly learning sentence embeddings and syntax with unsupervised tree-lstms. *Natural Language Engineering*, 25(4):433–449.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 216–223.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *TACL*, 6:253–267.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to compose words into sentences with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

A Appendices

A.1 Hyper parameters

Hyperparameters are fixed given literature on the domain, in particular regarding choices made in Tai et al. (2015). For all experiments detailed in Section 3, the batch size is fixed to 25, weight decay to $1e^{-4}$ and gradient clipping set to 5.0. The learning rate is set to 0.025 for the TREE-LSTM parameters. When using a pre-training procedure for the parser, we set the learning rate to $5e^{-3}$ and use the following warm-up: for the first epoch, the parser is frozen. For the following epochs, all parameters are trained. At each epoch the parser learning rate is divided by a factor of two. Wihtout pre-training, the learning rate is set to $5e^{-4}$ for the parser. All model weights are initialized with a Xavier distribution. The hidden size of the similarity architecture is set to 50. We used the adagrad optimizer.

A.2 Model Architecture

For models in Table 1, we use an architecture with only one layer for the LSTM and 300-dimensional GloVe embeddings.

Regarding the biaffine parser, all parameters are chosen given Dozat and Manning (2017) recommendations. We use a hidden size of 150 for the MLPs layers and 100 for the biaffine layer. The dropout rate is fixed to 0.33. We use an open-source implementation of the parser and replace the pos-tags features with character level features. Therefore we don't need pos-tags annotations to parse our corpus[5]. We encode words using 100-dimensional GloVe embedding and a character embedding size of 50. Word vectors are then fed to a bidirectional LSTM with 3 layers of size 400.

A.3 Visualization of the parses

We present the parses obtained using various configurations from our model. Regarding the parses obtained without any pre-training on the PTB, we observe all nodes tend to be connected to an arbitrarily chosen root.

In our low supervised setup, we observe the parser's fine-tuning enables the model to adjust the parse from Figure 2 into the structure represented in Figure 3. The obtained structure does match the one produced using more pretraining samples from the PTB dataset.

[5] https://github.com/yzhangcs/biaffine-parser

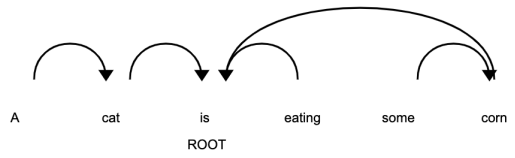

Figure 2: **Example of parse:** Parse obtained using the the **PTB-100** (×) configuration.

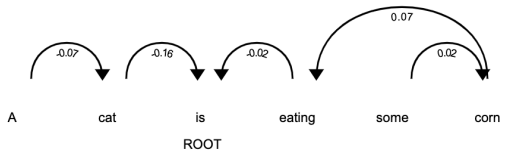

Figure 3: **Example of parse:** Parse obtained using the the **PTB-100** (✓) configuration. We include the weights α produced from the parser

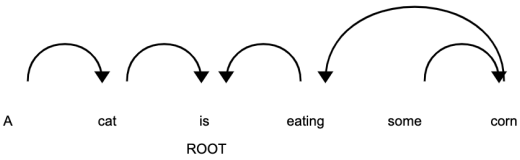

Figure 4: **Example of parse:** Parse obtained using the the **PTB-All** (×) configuration.

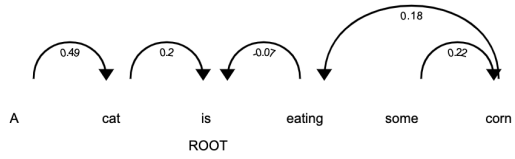

Figure 5: **Example of parse:** Parse obtained using the the **PTB-All** (✓) configuration. We include the weights α produced from the parser

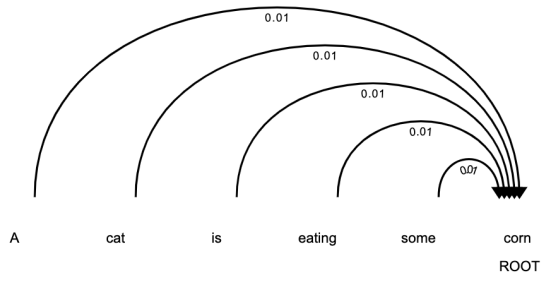

Figure 6: **Example of parse:** Parse obtained using the the **PTB-∅** (✓) configuration. We include the weights α produced from the parser

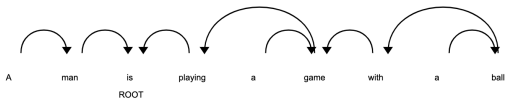

Figure 7: **Example of parse:** Parse obtained using the the **PTB-100 (×)** configuration.

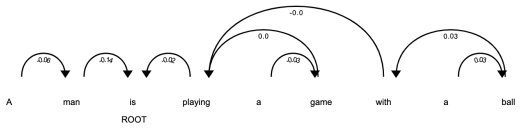

Figure 8: **Example of parse:** Parse obtained using the the **PTB-100 (✓)** configuration. We include the weights α produced from the parser

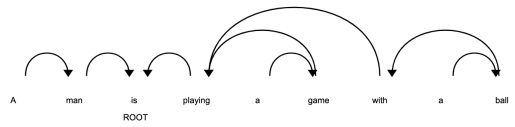

Figure 9: **Example of parse:** Parse obtained using the the **PTB-All (×)** configuration.

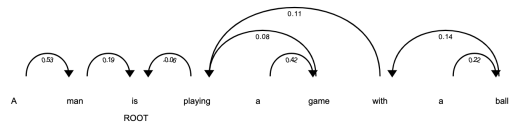

Figure 10: **Example of parse:** Parse obtained using the the **PTB-All (✓)** configuration. We include the weights α produced from the parser

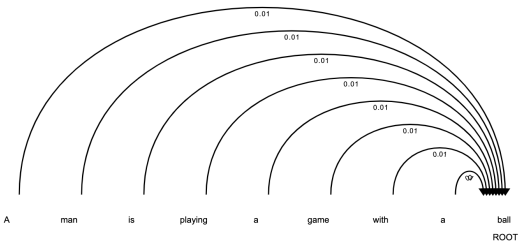

Figure 11: **Example of parse:** Parse obtained using the the **PTB-∅ (✓)** configuration. We include the weights α produced from the parser

==========
EACL 2021 Reviews for Submission #318
==========

Title: Unifying parsing and tree-structured models for generating sentence semantic representations
Authors: Antoine Simoulin and Benoit Crabbé
==========
META-REVIEW
==========

Comments: This paper proposes a method to induce a tree structure for a sentence, and learn its embedding based on the induced structure, with empirical results on a sentence classification task.

The reviewers agree that the paper presents detailed empirical results despite the length restrictions of a short paper. However, they are not convinced that the claims of the work are necessarily substantiated - in particular one claim about the differentiability of the method seems to be not adequately addressed. Moreover, experiments on a single task do not justify the generality of the method.

Taking into account all the above, I believe this paper might need more work before it is ready for publication.

==========
REVIEWER #1
==========

What is this paper about, what contributions does it make, what are the strengths and weaknesses, what are the questions for the authors?

The paper present a method for creating sentence embeddings based on syntactic structures.
The authors tested the method on the SICK dataset and obtained reasonable performance, even though they did not compare their model to previous work.

There is also an analysis about the amount of syntactic supervision needed to perform well on the task.
I have troubles though to understand what the authors mean with fine tuning the parser with the task loss.
The authors write that they take the score of each arc from a MST operation, but MST is not differentiable, but yet they claim that the approach is differentiable.
Other papers attempted to do what the authors tries to do, but they have used either relaxation https://arxiv.org/pdf/1906.09992.pdf or reinforcement learning https://arxiv.org/pdf/1902.09393.pdf, to overcome the problem of non differentiability.

What I believe is happening is that there is no update in the parser at all, this is also reflected in the results in table 2, where there is basically no difference between the frozen parser and fine tuned one. A way to check this would be to show the parser UAS on PTB before and after the fine tuning.

If the parser is actually updated with the task loss, I would ask the authors to be more clear in the model description, and explain how they make MST differentiable or how they avoid it.

Reasons to accept

The paper is a nice contribution for a short paper.

Reasons to reject

I think the claim the paper makes about unifying parsing and creating sentence representation is either unclear, or not correct.

Reviewer's Scores

Overall Recommendation: 2

==========
REVIEWER #2
==========

What is this paper about, what contributions does it make, what are the strengths and weaknesses, what are the questions for the authors?

This paper presents a sentence encoder model that benefits from an inductive bias from the additional syntactic parsing objective, which is optimized jointly or as a pre-training for the targeted downstream task. The proposed approach is built on a graph-based bi-affine model (Dozat and Manning, 2017) as a parser, and tree LSTM function (Tai et al, 2015) to compute a compositional representation of sentences. The authors show the proposed approach can be as effective as BERT model on a textual entailment dataset SICK, even without an external parser at test time.

Typos, Grammar, Style and References

Missing reference:
1. Bowman et al. 2016. A Fast Unified Model for Parsing and Sentence Understanding.
2. Liu et al. 2018. Structured Alignment Networks for Matching Sentences.

Typo(s):
- Line 247: ressources

Reasons to accept

Incorporating structural inductive bias to language understanding models might help models to leverage the signal from training examples that require syntactic analysis to make correct predictions. The direction that this paper pursues remains an interesting challenge for the community to achieve more robust and transparent inference models.

Reasons to reject

The positioning of the paper makes it hard to judge clearly the main contributions of the paper. The claim that the paper introduces a novel unified parser and semantic understanding model is hardly corroborated, since it lacks the comparison and discussion about its difference with existing models with a similar goal, such as SPINN (Bowman et al, 2016).

The analysis of the importance of structural biases on semantic tasks also will not be a sufficient contribution since the authors only evaluate on a single NLI dataset. SICK is limited in size, domain, and variety of inference types. Drawing conclusions based only on this result might be a bit of a stretch. Furthermore, the claim that syntactic supervision is needed to produce interpretable parses may be at odds with this finding from Liu et al (2018), which show that supervision from text-matching task could be used to infer a latent tree structure of the sentence input. More discussion on this would be interesting for the readers.

Reviewer's Scores

Overall Recommendation: 2.5

==========

REVIEWER #3

==========

What is this paper about, what contributions does it make, what are the strengths and weaknesses, what are the questions for the authors?

This paper presents a tree-structured model that performs parsing and the prediction of sentence embedding in a joint way. The model was evaluated on a textual entailment dataset (SICK) and shown to be effective compared to other current standard models.

Strengths:
- The paper is well written. Detailed experiments and analyses are presented, though there are limitations for short papers. The comparison between three settings, PTB-All, PTB-100, and PTB-empty) is interesting and informative.

Weaknesses:
- It would be helpful to show some examples of tasks (in particular, sentences involving such compositional structures as mentioned in 183-184), as well as error analyses.

The task here is called the "textual entailment task", but is different from the classification task (using three values, entailment, neutral, and contradiction). It would be better to call it the "semantic textual similarity (STS)" task, rather than the "textual entailment task".

Reasons to accept

See the "strengths" above.

Reasons to reject

See the "weaknesses" above.

Reviewer's Scores

Overall Recommendation: 3