Kalpa Publications in Computing

Volume 14, 2023, Pages 83–87

Proceedings of V XoveTIC Conference. XoveTIC 2022

A System for Constituent and Dependency Tree Linearization

Diego Roca, David Vilares, and Carlos Gómez-Rodríguez

Universidade da Coruña, CITIC, A Coruña, Spain
d.roca1@udc.es, david.vilares@udc.es, carlos.gomez@udc.es

Abstract

In this work, we introduce a framework that unifies existing implementations for the tasks of constituent and dependency parsing as sequence labeling problems. The system provides a way to encode both formalisms as sequences of one label per word, so they can be used with any existing general-purpose sequence labeling architecture. More particularly, we implement three linearizations to encode constituent trees and four linearizations for dependency trees. All encoding functions ensure completeness and injectivity. We will also train a sequence labeling neural system to learn such encodings, and compare their effectiveness on standard constituent (PTB and SPMRL treebanks) and dependency parsing (a subset of treebanks from the UD collection) evaluation frameworks.

1 Introduction

Parsing is a natural language processing (NLP) task in which, given an input raw sentence, a system automatically produces an output representing its syntactic structure. The most popular and widely adopted formalisms to represent such structure are constituent and dependency formalisms. On the one hand, the goal of constituent parsing is to obtain the syntactic structure of a sentence as a phrase structure tree, a derivative from the *subject-predicative* division from Greek and Latin. On the other hand, dependency parsing aims to generate a parse tree as a set of binary directed relations between words called dependencies that describe the syntactic roles (e.g., *direct object*, *subject*, *adverbial modifier*) that participate in the sentence. Although both formalisms present differences, they also share limitations, in particular the limited speed of the models used to obtain these syntactic representations. Transforming both of these problems into a sequence labeling task provides an advantage in efficiency while keeping competitive accuracy, and allows such representations to be learned by any black-box sequence labeling neural system. Several tree linearizations have been proposed in the last few years to cast these tasks as sequence labeling [3, 10, 6, 7] . However, there is no common suite that facilitates their use under an integrated framework. In this work, we design such a suite. More particularly, our system will allow the user to: (i) encode gold or predicted trees according to the desired syntactic formalism and a number of configuration options, (ii) decode linearized trees into its original representation, and (iii) train a neural sequence labeling system to perform parsing and measure performance in terms of accuracy and speed.

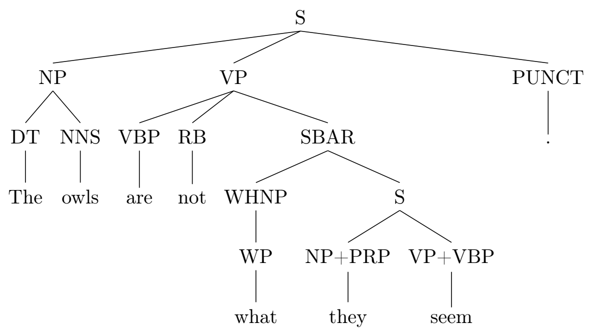
2 Constituent Parsing as Sequence Labeling

Let $w = [w_1, w_2, ..., w_{|w|}]$ be an input sequence of words and let $Tc_{|w|}$ be the set of possible constituent trees with $|w|$ leaf nodes. To cast the task of constituent parsing as a sequence labeling task we must define a set of labels Lc that allows us to encode each tree in $Tc_{|w|}$ as a unique sequence of labels $Lc^{|w|}$ and an encoding function $Fc_{|w|} : Tc_{|w|} \rightarrow Lc_{|w|}$ that allows us to translate the trees into those labels. The labels in Lc will be shaped as a 3-tuple $lc_i = (nc_i, lc_i, uc_i)$ where lc_i is the last common ancestor between w_i and w_{i+1}, uc_i is (if exists) the *collapsed leaf unary chain*[3] of w_i and nc_i is the encoded value of the number of common ancestors. For the nc_i field, three different encodings are proposed:

1. $Fc^{abs}_{|w|}$ [3]: Encodes nc_i as the number of common ancestors between the words w_i and w_{i+1} (see nc^{ABS}_i in figure 1).

2. $Fc^{rel}_{|w|}$ [3]: Encodes nc_i as the difference with respect to the number of common ancestors encoded in nc_{i-1}. This considerably reduces the sparsity of the output vocabulary space (see nc^{REL}_i in figure 1).

3. $Fc^{dyn}_{|w|}$ [10]: Encodes nc_i using $Fc^{abs}_{|w|}$ depending if its value is inside a range empirically defined, and as $Fc^{rel}_{|w|}$ otherwise. This way, it takes advantage both of (i) $Fc^{rel}_{|w|}$ performing better for sentences with deeper constituent trees, and (ii) $Fc^{abs}_{|w|}$ performing better for shallower constituent trees (see nc^{DYN}_i in figure 1).



(a)

w_i	nc^{ABS}_i	nc^{REL}_i	nc^{DYN}_i
The	2	2	2
owls	1	-1	1
are	2	1	2
not	2	0	2
what	3	1	1
they	4	1	1
seem	1	-3	1
.	1	0	1

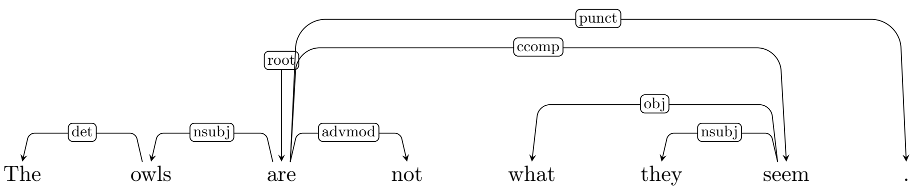
(b)

Figure 1: Example of a constituent tree with collapsed unary branches (a) and how the nc_i field is encoded using the naive absolute (nc^{ABS}_i, red), naive relative (nc^{REL}_i, blue) and dynamic encodings (nc^{DYN}_i).

3 Dependency Parsing as Sequence Labeling

Let $Td_{|w|} = (V, A)$ be a dependency tree for a sentence $w = [w_1, ..., w_{|w|}]$ where $i \in V$ represents the position of the word w_i, and A is the set of binary relations between indexes, such that each $a \in A$ has the form $a = (h, r, d)$ where: $h \in V$ represents the head of the relation, $d \in V$ represents the dependant of the relation, and $r \in R$ represents the type of relation that exists among them such that R is a set of syntactic functions (e.g. subject, or direct object). To cast the task of dependency parsing as a sequence labeling task we must define a set of labels Ld that allows us to encode $Td_{|w|}$ as a unique sequence of labels $Ld^{|w|}$, and an encoding function $Fd_{|w|} : Td_{|w|}| \rightarrow Ld_{|w|}$ that allows us to translate the trees into those labels. All the encodings implemented deal with the conversion of the set of dependency edges corresponding to a sentence w to a sequence of labels shaped as $ld_j = (x_j, r_j)$, where $r_j \in R$ is the corresponding dependency type to word w_j and x_j is the encoding-specific [6] value that defines how a given arc is encoded:

1. $Fd^{abs}_{|w|}$: Encodes x_j as the value of h_j in a_j (see x^{ABS}_j in figure 2).
2. $Fd^{rel}_{|w|}$: Encodes x_j as the difference of $h_j - d_j$ in a_j (see x^{REL}_j in figure 2).
3. $Fd^{pos}_{|w|}$: Encodes x_j as a tuple (p_j, o_j) where p_j indicates the part-of-speech (POS) tag at position h_j and o_j indicates (i) the direction from where the head is (being to the left if $o_j < 0$ or to the right if $o_j > 0$), and (ii) the number of POS tags p_j found in the sentence until the correct one (see x^{POS}_j in figure 2).
4. $Fd^{brk}_{|w|}$: Encodes x_j as a string defined by the regular expression (<)?((\)*|(/)*)(>)? where the presence of each character in w_j indicates an outgoing / incoming dependency arrow (see x^{BRK}_j in figure 2). We also implemented modifications $Fd^{brk-2pg}_{|w|}$ and $Fd^{brk-2pp}_{|w|}$ to deal with non-projective trees, following the variations introduced in [7].



w_i	The	owls	are	not	what	they	seem	.
i	1	2	3	4	5	6	7	8
POS	DT	NNS	VBP	RB	WP	PRP	VBP	PUNCT
t_j	det	nsubj	root	advmod	obj	nsubj	ccomp	punct
x^{ABS}_j	2	3	0	3	7	7	3	3
x^{REL}_j	+1	+1	-3	-1	2	+1	-4	-5
x^{POS}_j	(+1,NNS)	(+1,VBP)	(0,ROOT)	(-1,VBP)	(+1,VBP)	(+1,VBP)	(-1,VBP)	(-2,VBP)
x^{BRK}_j		<\	<\	///>		<	<\\>	>

Figure 2: Encoding of a given dependency tree with the x_j field encoded according to the naive absolute, naive relative, POS-based, and bracketing-based encodings.

4 Results and Conclusions

We check the learnability of the encodings by training the MACHAMP [9] neural sequence labeling model. Two sets of experiments were run: (i) with the inclusion of multilingual BERT [2] embeddings and (ii) with the inclusion of BERT embeddings and multi-task learning [1] capabilities to predict the part-of-speech tags too. The treebanks employed are the English Penn Treebank [4] and SPMRL[1] [8] for constituent parsing, and Universal Dependencies [5] for

[1]SPMRL includes Basque, French, German, Hebrew, Hungarian, Korean, Polish and Swedish languages

dependency parsing (for better comparison, we use the same languages for both formalisms). See tables 1 and 2 for constituent and dependency parsing results on the test sets.

Based on the experimental results, for constituent parsing the best results across the board are obtained by the dynamic encoding. For dependency parsing there is more diversity, bracketing-based encodings seem to perform overall better, although simpler encodings such as the relative positional encoding still perform robustly. Finally, the use of multi task learning does not come with consistent improvements; more experiments are required to better understand the reasons for this behavior, and we leave this as future work.

Tool	Enc	Eng	Basq	Fr	Ger	Hebr	Hun	Kr	Po	Sw
MACHAMP^{BERT}	$Fc^{abs}_{\|w\|}$	92.11	78.16	**90.80**	89.61	87.92	**93.57**	**88.14**	95.05	76.81
	$Fc^{rel}_{\|w\|}$	92.23	80.38	90.04	88.86	88.65	93.23	87.21	94.79	79.27
	$Fc^{dyn}_{\|w\|}$	**93.35**	**80.93**	90.38	**89.99**	**88.92**	93.47	87.60	**95.55**	**80.95**
MACHAMP$^{BERT}_{MTL}$	$Fc^{abs}_{\|w\|}$	92.87	75.23	89.00	87.12	87.36	91.61	85.68	93.40	74.45
	$Fc^{rel}_{\|w\|}$	92.51	78.25	87.10	88.45	88.39	90.63	85.03	92.86	75.95
	$Fc^{dyn}_{\|w\|}$	93.09	79.15	87.36	88.66	88.63	91.29	85.58	94.47	79.49

Table 1: F-Score (higher is better) for constituent parsing on the test sets of the PTB and SPMRL treebanks.

Tool	Enc	Eng	Basq	Fr	Ger	Hebr	Hun	Kr	Po	Sw
MACHAMP^{BERT}	$Fd^{abs}_{\|w\|}$	83.53	87.82	84.53	95.21	66.70	35.82	69.99	97.14	66.24
	$Fd^{rel}_{\|w\|}$	85.92	88.90	85.92	**97.53**	82.68	62.23	72.95	**98.55**	78.26
	$Fd^{pos}_{\|w\|}$	86.42	87.56	83.42	89.96	83.51	**75.68**	51.33	92.11	78.69
	$Fdbrk_{\|w\|}$	88.20	90.86	88.07	96.27	**89.17**	74.42	77.07	97.58	83.83
	$Fd^{brk-2pg}_{\|w\|}$	**88.34**	**91.07**	88.20	97.11	89.03	73.91	**77.29**	97.56	**83.88**
	$Fd^{brk_2pp}_{\|w\|}$	**88.34**	90.58	**88.34**	97.25	89.06	73.89	76.57	97.88	83.74
MACHAMP$^{BERT}_{MTL}$	$Fd^{abs}_{\|w\|}$	78.93	81.82	73.93	93.63	61.93	20.74	58.83	95.98	51.37
	$Fd^{rel}_{\|w\|}$	82.92	86.23	82.14	93.48	77.20	51.45	71.52	96.12	71.61
	$Fd^{pos}_{\|w\|}$	85.73	86.93	81.92	88.16	74.12	71.82	51.12	92.30	77.71
	$Fd^{brk_d}_{\|w\|}$	86.41	89.96	86.31	95.02	85.03	58.88	76.19	96.13	78.11
	$Fd^{brk-2pg}_{\|w\|}$	86.12	89.12	86.07	95.11	84.87	67.23	76.06	96.02	78.05
	$Fd^{brk-2pp}_{\|w\|}$	86.36	89.66	86.13	95.07	82.05	67.24	76.18	96.11	77.89

Table 2: Labeled attachment score (higher is better) for the dependency parsing encodings on the test sets of the UD treebanks used in this work.

Conclusion In this work, we present a suite that includes several linearization algorithms to encode constituent and dependency trees as a sequence of labels, that can be learned by any generic sequence labeling system. The results on standard evaluation frameworks show that such sequence labeling models obtain competitive results, and that the way in which the trees are encoded plays an important role.

References

[1] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Carlos Gómez-Rodríguez and David Vilares. Constituent parsing as sequence labeling. *arXiv preprint arXiv:1810.08994*, 2018.

[4] Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273, 1994.

[5] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*, 2020.

[6] Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. Viable dependency parsing as sequence labeling. *arXiv preprint arXiv:1902.10505*, 2019.

[7] Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. Bracketing encodings for 2-planar dependency parsing. *arXiv preprint arXiv:2011.00596*, 2020.

[8] Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. Statistical parsing of morphologically rich languages (spmrl) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, 2010.

[9] Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. Massive choice, ample tasks (machamp): A toolkit for multi-task learning in nlp. *arXiv preprint arXiv:2005.14672*, 2020.

[10] David Vilares, Mostafa Abdou, and Anders Søgaard. Better, faster, stronger sequence tagging constituent parsers. *arXiv preprint arXiv:1902.10985*, 2019.