Knowledge-Based Systems 236 (2022) 107760

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multitask Pointer Network for multi-representational parsing

Daniel Fernández-González *, Carlos Gómez-Rodríguez

Universidade da Coruña, CITIC, FASTPARSE Lab, LyS Group, Depto. de Ciencias de la Computación y Tecnologías de la Información, Campus de Elviña, s/n, A Coruña, 15071, Spain

ARTICLE INFO

Article history:
Received 4 June 2021
Received in revised form 20 September 2021
Accepted 14 November 2021
Available online 26 November 2021

Keywords:
Natural language processing
Computational linguistics
Parsing
Dependency parsing
Constituent parsing
Neural network
Deep learning

ABSTRACT

Dependency and constituent trees are widely used by many artificial intelligence applications for representing the syntactic structure of human languages. Typically, these structures are separately produced by either dependency or constituent parsers. In this article, we propose a transition-based approach that, by training a single model, can efficiently parse any input sentence with both constituent and dependency trees, supporting both continuous/projective and discontinuous/non-projective syntactic structures. To that end, we develop a Pointer Network architecture with two separate task-specific decoders and a common encoder, and follow a multitask learning strategy to jointly train them. The resulting quadratic system, not only becomes the first parser that can jointly produce both unrestricted constituent and dependency trees from a single model, but also proves that both syntactic formalisms can benefit from each other during training, achieving state-of-the-art accuracies in several widely-used benchmarks such as the continuous English and Chinese Penn Treebanks, as well as the discontinuous German NEGRA and TIGER datasets.

1. Introduction

Numerous artificial intelligence systems that demand natural language processing of texts and speech are currently using syntactic formalisms for representing the grammatical structure of sentences. Among them, we can find those that recently use it for machine translation [1–3], relation and event extraction [4], opinion mining [5,6], question answering [7,8], sentence classification [3], sentiment classification [9], summarization [10] or semantic role labeling and named entity recognition [11]. To that end, two widely-known formalisms are commonly used: *constituent* and *dependency* representations.

Constituent trees, which are commonly used in tasks where span information is crucial, represent the syntax of a sentence by means of constituents (also called *phrases*) that hierarchically and from the bottom up group words and/or subtrees located in lower levels. We can find two kinds of constituent trees: *continuous* and *discontinuous* (described in Fig. 1(a) and (d), respectively). The latter extends the former by allowing constituents with discontinuous spans, which results in phrase-structure trees with crossing branches. These are necessary for describing some wh-movement, long-distance extractions, dislocations, cross-serial dependencies and other linguistic phenomena common in free word order languages such as German [12].

On the other hand, in a dependency tree each word of the sentence is attached to another by a directed link that describes a dependency relation between that word and its parent (also called *head*). This structure is known for representing information closer to semantic relations and can be classified as *projective* or *non-projective* (depicted in Fig. 1(c) and (f), respectively). Non-projective dependency trees allow crossing dependencies, and can model the same linguistic phenomena described by discontinuous constituent trees.

Since the information described in a constituent tree is not fully encoded into a dependency tree and vice versa [13], typically parsers are exclusively trained to produce either dependency or constituent structures and, in some cases, restricted to the less complex continuous/projective representations.

There are a few exceptions, i.e., approaches trained to generate both constituents and dependencies. For instance, the chart parser of Zhou and Zhao [14] generates continuous and projective structures with a single $O(n^5)$ model, and the sequence labeling parser of Strzyz et al. [15] combines continuous constituents with non-projective dependency structures.[1] In both cases, which are discussed in more detail in Section 5, representations are shown to benefit each other in terms of accuracy.

* Corresponding author.
E-mail addresses: d.fgonzalez@udc.es (D. Fernández-González), carlos.gomez@udc.es (C. Gómez-Rodríguez).
URLs: https://fastparse.grupolys.org (D. Fernández-González), http://www.grupolys.org/~cgomezr/ (C. Gómez-Rodríguez).

[1] As explained in Section 5, parsers based on lexicalized grammars were also trained on both structures in the pre-deep-learning era.

https://doi.org/10.1016/j.knosys.2021.107760

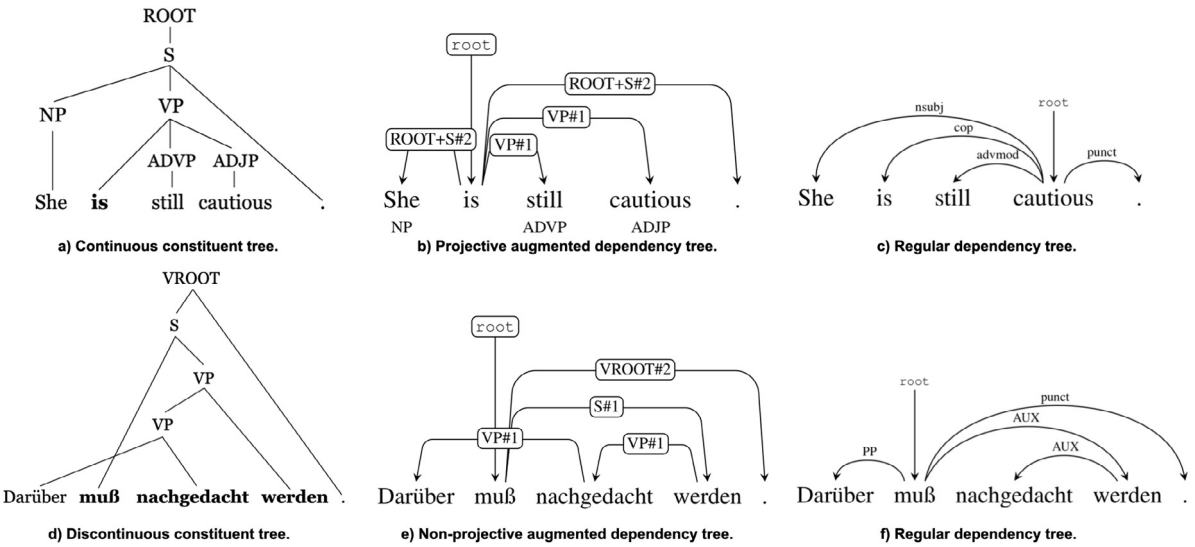

Fig. 1. Constituent, augmented and regular dependency representations of continuous English and discontinuous German sentences. Head words of constituent trees are marked in bold. Please note that regular and augmented dependency trees differ since, while head words are marked following a syntactic strategy in the augmented variant, in regular dependency trees head words are indicated according to a semantic approach.

However, to our knowledge, no such joint training approaches have been defined that support both non-projective dependency trees and discontinuous constituents; and the most accurate and least computationally complex models for these formalisms are single-representation approaches: graph-based [16] and transition-based [17,18] models for non-projective dependencies, and transition-based parsers [19–21] for discontinuous phrase-structure trees.

In order to fill this gap, we propose a novel multitask transition-based parser that can efficiently generate unrestricted constituent and dependency structures (i.e., discontinuous constituents and non-projective dependencies, although it can also be restricted to continuous/projective structures if desired) from a single trained model. We design an encoder–decoder neural architecture that is jointly trained across the syntactic information represented in the two formalisms by following a multitask learning strategy [22]. Inspired by Fernández-González and Gómez-Rodríguez [21], we model constituent trees as augmented dependency structures [23] and use two separate task-specific decoders to produce both regular and augmented dependency trees. Each decoder implements a Pointer Network [24] and a multi-class classifier [16] to incrementally produce labeled dependencies from left to right, as proposed by Fernández-González and Gómez-Rodríguez [18]. Finally, the decoding runtime ($O(n^2)$) and the required memory space of our multi-representational approach remains the same as the single-task dependency parser by Fernández-González and Gómez-Rodríguez [18], since a single model is trained and the multitask learning strategy has no impact on decoding time, allowing both decoders to be run in parallel.

We test our multi-representational neural model[2] on the continuous English and Chinese Penn Treebanks [25,26] and on the discontinuous NEGRA [27] and TIGER [28] datasets. In all benchmarks, our approach outperforms single-task parsers [18,21], which proves that learning across regular dependency trees and constituent information (encoded in dependency structures) leads to gains in accuracy in both tasks, obtaining competitive results in all cases and surpassing the current state of the art in several datasets.

[2] Source code available at https://github.com/danifg/MultiPointer.

The remainder of this article is organized as follows: Section 2 introduces the constituent-to-dependency encoding technique developed by Fernández-González and Martins [23]. In Section 3, we describe in detail the proposed multitask Pointer Network architecture. In Section 4, we extensively evaluate the proposed neural model on continuous/projective and discontinuous/non-projective treebanks, as well as include a thorough analysis of their performance. Section 5 presents other research works that study the joint training of neural models across different syntactic formalisms. Lastly, Section 6 contains a final discussion.

2. Constituent trees as dependency structures

Since our multitask approach is based on the dependency parser by Fernández-González and Gómez-Rodríguez [18], constituent trees must be represented as dependencies in order to be processed. This was recently explored for neural discontinuous constituent parsing in [21] by using the encoding by Fernández-González and Martins [23]. In this work, we extend it to continuous phrase-structure datasets, where the non-negligible frequency of unary nodes requires additional processing.

2.1. Preliminaries

Let $w_1, w_2, \ldots, w_n$ be a sentence and w_i the word at position i. A constituent tree is defined by constituents (as internal nodes) hierarchically organized over these n words (as leaf nodes). Each phrase (or constituent) is defined as a tuple $(X, \mathcal{S}, w_h)$ that includes a non-terminal symbol X, the set of words w_i included in its span ($\mathcal{S}$); and, w_h, the word in $\mathcal{S}$ that acts as head and that can be marked by using a language-specific handwritten set of rules. For example, the head word of constituents S and VP in Fig. 1(a) is the word *is*. Furthermore, we say that a constituent tree is *continuous* if there are no constituents whose yield $\mathcal{S}$ is a discontinuous substring of the sentence. If this does not hold, the tree is classified as *discontinuous*, and then there is at least one constituent with one or more gaps in its span. For instance, the word *muß* interrupts the span of constituent (VP, {*Darüber*, *nachgedacht*}, *nachgedacht*) in Fig. 1(d), resulting in a phrase structure with crossing branches. Finally, constituents with exactly one child are known as *unary constituents* (for instance, ROOT, NP, ADVP and ADJP in Fig. 1(a)).

Unlike constituent structures, dependency trees do not require extra internal nodes and are exclusively composed of the words w_i of the sentence (plus an artificial `root` node) and binary directed links to connect them. Each dependency link is represented as (w_h, w_d, l), where w_h is the head word of the dependent word w_d ($h, d \in [1, n]$) and l a dependency label. Additionally, a dependency tree is classified as *projective* if we can find a directed path from w_h to all words w_i between words w_h and w_d for every dependency link (w_h, w_d, l). If this does not hold, it is considered a *non-projective* dependency tree, as the one with crossing arcs depicted in Fig. 1(e).

2.2. Constituent-to-dependency conversion

Fernández-González and Martins [23] designed an encoding technique to represent a unariless constituent tree with m words as a set of $m - 1$ labeled dependency arcs with enriched information (plus an arc from `root`), where discontinuous phrase structures are encoded as non-projective dependency trees and continuous structures as projective trees, as shown in Fig. 1(b) and (e) for constituent trees in Fig. 1(a) and (d), respectively. To that end, for each constituent $(X, \mathcal{S}, w_h)$ with head word w_h, each child node w_d (different from w_h) is encoded into the unlabeled dependency link (w_h, w_d). Please note that a constituent's non-head child nodes w_d might be a word or another constituent $(Y, \mathcal{G}, w_d)$ with w_d as head word. Additionally, these dependencies are augmented with an arc label that includes the non-terminal name X concatenated with an index k that indicates the hierarchical order in which non-terminal nodes are built in the tree, resulting in labeled dependency arcs with the form $(w_h, w_i, X\#k)$. Index k was included for those cases where several constituents share the same head word, but they are placed in the tree at a different level. For instance, constituent (S, {*Darüber*, *muß*, *nachgedacht*, *werden*}, *muß*) in Fig. 1(d) is represented as the augmented dependency arc (*muß*, *werden*, S#1) in Fig. 1(e); and constituent (VROOT, {*Darüber*, *muß*, *nachgedacht*, *werden*, .}, *muß*) is encoded as (*muß*, ., VROOT#2). Both share head word *muß*, but the latter is built on top of the former and this must be encoded by hierarchical orders 1 and 2; otherwise, after the deconversion, the resulting structure would be a single constituent (named S or VROOT) that spans all the sentence.

Finally, unary constituents are not directly supported by this encoding strategy. While Fernández-González and Martins [23] proposed to remove all unary nodes and recover them in a post-processing step, we decided to incorporate unary constituents into the resulting augmented dependency tree by collapsing non-leaf unary chains (for instance, ROOT from Fig. 1(a) into ROOT+S) and saving leaf unary nodes lost after the encoding by assigning them to words (as can be seen in Fig. 1(b) for NP, ADVP and ADJP).

2.3. Constituent trees recovery

The original unariless constituent trees can be decoded from augmented dependency trees by, following a post order traversal, building constituents from the set of dependencies composed of each head word together with its dependents and following the hierarchical order dictated by the index k and non-terminal name X encoded into each of the dependency labels. Due to erroneous predictions, it might be the case that heads or dependency labels are mistakenly assigned in the resulting augmented dependency tree; however, Fernández-González and Martins [23]'s technique guarantees that the output is a well-formed constituent tree (which, of course, will differ from the gold tree). For instance, imagine that the word *cautious* in Fig. 1(b) is erroneously attached to the word *still* (instead of being connected to the verb *is*), then, instead of a single flat VP with three child nodes, the resulting constituents would be two VPs (the first would have as child nodes the word *is* and a second VP, which would group the words *still* and *cautious*). We can also find different scenarios where dependency labels are erroneously predicted, requiring ad-hoc heuristics during the recovery to deal with some inconsistencies:

- *Same indices, but different non-terminal names:* Note that dependency labels with the same head and at the same level (same index k) should share the same non-terminal name so that a flat constituent can be recovered. If this does not hold, then the dependency label of the dependent closer to the head will be the one chosen for tagging the resulting constituent. For instance, if the arcs *is*→*still* and *is*→*cautious* in Fig. 1(b) were tagged with labels VP#1 and (incorrect) NP#1, respectively; then we would use non-terminal label VP for naming the output flat constituent and NP would be discarded. Alternatively, we could consider that the non-terminal name is correct and index k was wrongly predicted: in our running example, we might think that the non-terminal name of dependency label NP#1 is correct, but the resulting constituent NP should be in a higher level (with the correct label being NP#2, for instance). This heuristic would lead us to build a constituent NP with a nested VP. However, Fernández-González and Martins [23] decided to follow a more conservative strategy that tends to produce flatter structures.
- *Non-nested indices in continuous parsing:* When the reverse conversion is restricted to continuous constituent trees, erroneous dependency labels might lead to discontinuous structures (even when the augmented dependency tree is projective). For example, if the arcs *is*→*still* and *is*→*cautious* were tagged with labels (incorrect) VP#2 and VP#1, respectively; then the resulting constituent would be a discontinuous constituent VP with two child nodes: the word *still* and a non-nested VP (with a discontinuous yield composed of the words *is* and *cautious*). This would be a well-formed phrase-structure tree in a discontinuous scenario; however, to produce continuous structures, hierarchical indices of dependent words closer to the head should always be the same or lower than adjacent and more distant siblings, thus ensuring that flat or nested continuous constituents will be obtained. In our running example, if the arcs *is*→*still* and *is*→*cautious* were erroneously tagged with labels VP#2 and VP#1, respectively; then we would decrease index 2 of dependent word *still* until reaching the index of the adjacent and more distant sibling (the word *cautious*). In this case, the index is set to 1 and a flat constituent VP with three child nodes is built.

With respect to unary recovery, it is worth noting that, while Penn treebanks present a significant amount of unary constituents, they are very uncommon in discontinuous datasets: NEGRA has no unaries at all and TIGER contains less than 1%. Therefore, we only perform unary recovery in Penn treebanks. To do so, we simply uncollapse unary chains encoded in dependency labels and, for recovering leaf unary nodes lost after the encoding, we use a tagger in a post-processing step. More in detail, we employ the neural sequence tagger developed by Yang and Zhang [29] for assigning to each word a possible leaf unary node (or a sequence of unaries collapsed into a single tag) seen in the training dataset or the tag `NONE` (if there is no unary node above that word).

2.4. Regular vs. augmented dependency trees

Although both the constituent-based and regular dependency structures are directed trees of n nodes, each provides exclusive information: span phrase information is included in arc

labels of the augmented variants, and regular dependency labels provide additional semantic information not described in phrase-structure trees. Furthermore, regular dependency trees differ from augmented ones, not only in the label set, but also in the conversion process. Although dependency trees are often generated from constituent trees, different head-rule sets for marking head words and other transformations can be applied in that process. While a set of syntactic rules are used for identifying head nodes when augmented dependency trees are produced, a semantic-based transformation is applied for choosing the semantic heads necessary for generating regular dependency structures. This is the reason why dependency structures in Fig. 1(b) and (e) are different from Fig. 1(c) and (f), respectively: for the English example, we use the head-rule set by Collins [30] in our constituent-to-dependency encoding, while regular dependency trees were obtained following the Stanford Dependencies conversion [31]; and, for the German sentence, the augmented dependency tree requires a non-projective structure to fully encode the discontinuous constituent tree, while the regular dependency tree represents the syntax (and semantics) of the sentence with just a projective structure. This will train the parser across a broader variety of syntactic representations and notations.

3. Multitask neural architecture

To develop a neural network capable of producing state-of-the-art, unrestricted constituent and dependency parses, we join two transition-based parsers recently presented under the same architecture: Fernández-González and Gómez-Rodríguez [18] for non-projective dependency parsing, and Fernández-González and Gómez-Rodríguez [21], an extension of the former that can produce discontinuous constituent trees. As explained before, we additionally extend the latter to also deal with continuous phrase structures and unary constituents.

Fernández-González and Gómez-Rodríguez [18] relies on *Pointer Networks* [24] to perform unlabeled dependency parsing. After learning the conditional probability of a sequence of numbers that represent positions from the input, these neural networks use a mechanism of attention [32] to select those positions during decoding. Unlike regular sequence-to-sequence architectures, Pointer Networks do not require a fixed dictionary based on the whole training dataset, but the dictionary size is specifically defined by each input sequence length. Fernández-González and Gómez-Rodríguez [18] adapt Pointer Networks to implement a transition-based approach that, starting at the first word of a sentence of length n, sequentially attaches, from left to right, the current focus word to the pointed head word, incrementally building a well-formed dependency tree in just n steps. This can be also seen as a sequence of n SHIFT-ATTACH-p transitions, each of which connects the current focus word to the head word in the pointed position p, and then moves the focus to the next word. In addition, a *biaffine* classifier [16] jointly trained is used for predicting dependency labels.

Inspired by Fernández-González and Gómez-Rodríguez [18], we introduce a novel neural architecture with two *task-specific decoders*: each word of the input sentence is attached to its regular head by the first decoder, and to its augmented dependency head by the second decoder. Additionally, each decoder provides a biaffine classifier trained on its task-specific label set. Since both decoders are aligned, the resulting system requires just n steps to dependency and constituent[3] parse a sentence of length n, easily allowing joint training.

More specifically, our neural architecture is composed of:

Shared encoder. Each input sentence $w_1, \ldots, w_n$ is encoded by a BiLSTM-CNN architecture [33], word by word, into a sequence of *encoder hidden states* $\mathbf{h}_1, \ldots, \mathbf{h}_n$. In particular, a Convolutional Neural Network (CNN) is used for extracting a character-level representation of words ($\mathbf{e}_i^c$) and this is concatenated with a word embedding ($\mathbf{e}_i^w$) to create the vector representation $\mathbf{x}_i$ for each input word w_i. Additionally, POS tag embeddings ($\mathbf{e}_i^p$) are used when gold POS tags are available[4]:

$$\mathbf{x}_i = \mathbf{e}_i^c \oplus \mathbf{e}_i^w \oplus \mathbf{e}_i^p$$

Then, the word representation $\mathbf{x}_i$ is fed one-by-one into a BiLSTM for generating vector representations $\mathbf{h}_i$, which encode context information captured in both directions:

$$\mathbf{h}_i = \mathbf{h}_{li} \oplus \mathbf{h}_{ri} = \mathbf{BiLSTM}(\mathbf{x}_i)$$

Additionally,a special vector representation $\mathbf{h}_0$, denoting the ROOT node, is prepended at the beginning of the sequence of encoder hidden states.

Finally, we extend the encoder with deep contextualized word embeddings ($\mathbf{e}_i^{BERT}$) extracted from the pre-trained language model BERT [34] by directly concatenating them to the resulting basic word representation $\mathbf{x}_i$ before feeding the BiLSTM-based encoder:

$$\mathbf{x}'_i = \mathbf{x}_i \oplus \mathbf{e}_i^{BERT};\ \ \mathbf{h}_i = \mathbf{BiLSTM}(\mathbf{x}'_i)$$

Task-specific decoders. Each decoder d is implemented by a separate LSTM that, at each time step t, receives as input the encoder hidden state $\mathbf{h}_i$ of the current focus word w_i and generates a *decoder hidden state* $\mathbf{s}_t^d$ [5]:

$$\mathbf{s}_t^d = \mathbf{LSTM}_d(\mathbf{h}_i)$$

Additionally, a *pointer layer* is implemented for each decoder by an attention vector $\mathbf{a}_t^d$ to perform unlabeled parsing. This vector is generated by computing scores for all possible head-dependent pairs between the current focus word (represented by $\mathbf{s}_t^d$) and each word from the input (represented by encoder hidden representations $\mathbf{h}_j$ with $j \in [0, n]$). To that end, a scoring function based on the biaffine attention mechanism [16] is used and, then, a probability distribution over the input words is computed:

$$\mathbf{v}_{tj}^d = \mathbf{score}(\mathbf{s}_t^d, \mathbf{h}_j) = f_1(\mathbf{s}_t^d)^T W f_2(\mathbf{h}_j) + \mathbf{U}^T f_1(\mathbf{s}_t^d) + \mathbf{V}^T f_2(\mathbf{h}_j) + \mathbf{b};$$

$$\mathbf{a}_t^d = \mathbf{softmax}(\mathbf{v}_t^d)$$

where W is the weight matrix of the bi-linear term, $\mathbf{U}$ and $\mathbf{V}$ are the weight tensors of the linear terms, $\mathbf{b}$ is the bias vector and $f_1(\cdot)$ and $f_2(\cdot)$ are two single-layer multilayer perceptrons (MLP) with ELU activation [16].

Each attention vector $\mathbf{a}_t^d$ will serve as a pointer to the highest-scoring position p from the input, leading the parsing algorithm to create a dependency arc from the head word (w_p) to the current focus word (w_i). In case this dependency arc is forbidden since it generates cycles in the already-created dependency tree, the next highest-scoring position in $\mathbf{a}_t^d$ will be considered as output instead. Furthermore, the projectivity constraint is also enforced when processing continuous treebanks, discarding arcs that produce crossing dependencies. After the decoding process (where each word is attached to another word at each step), we obtain a

[3] Constituent trees are obtained after decoding resulting augmented dependency trees.

[4] As noticed by Ma et al. [17] and Fernández-González and Gómez-Rodríguez [21], the usage of predicted POS tags does not lead to gains in accuracy. Therefore, we only use POS tags in experimental settings where they are gold.

[5] Unlike [18], we do not use other encoder hidden states as extra feature information for the decoder, since we noticed that practically the same accuracy can be achieved with this simple framework.

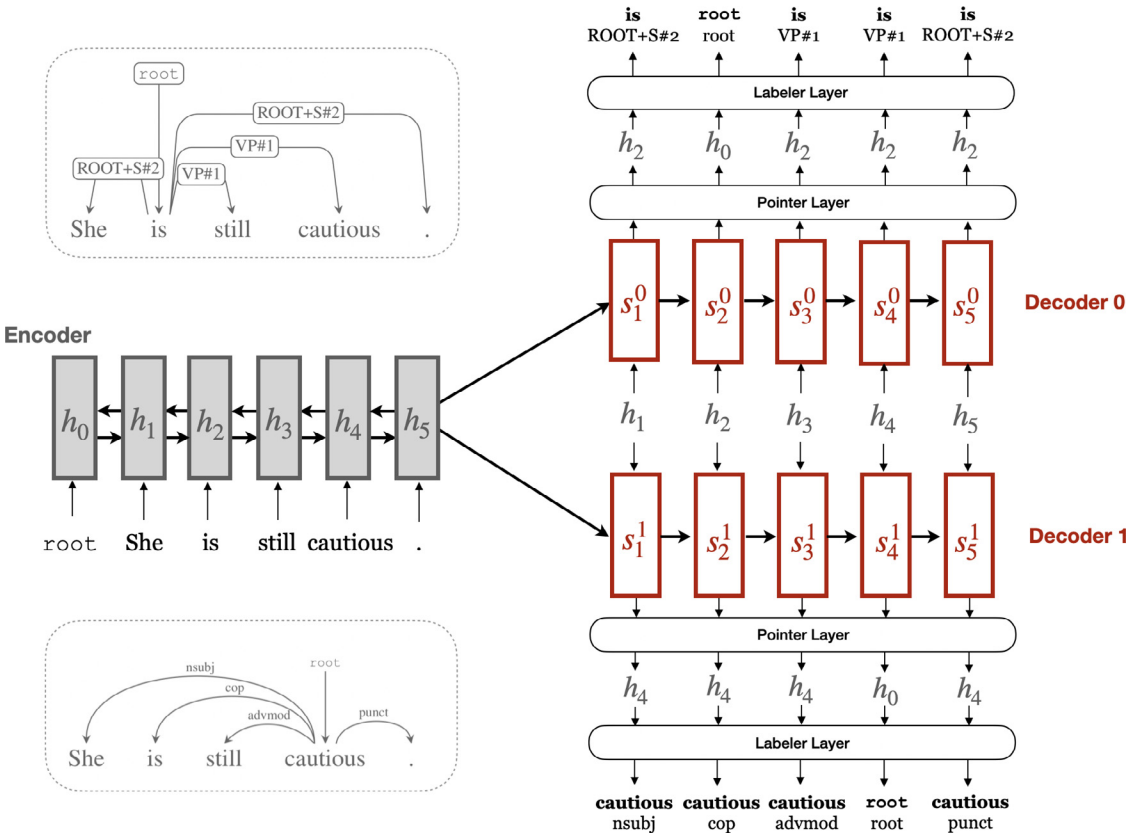

Fig. 2. Simplified sketch of our multitask neural architecture and decoding steps to parse the sentence in Fig. 1(a). `Decoder 0` and `Decoder 1` perform constituent-based and regular dependency parsing, respectively.

well-formed dependency tree where each word has a single head (except the artificial ROOT node that was not processed), with no cycles and, as a consequence of satisfying both the single-head and acyclicity constraints, all words are guaranteed to be connected.

Finally, each decoder trains a *labeler layer* (implemented as a multi-class classifier) to predict arc labels and produce a labeled dependency tree. In particular, after the pointer layer attaches the current focus word w_i (represented by $\mathbf{s}_t^d$) to the pointed head word w_p in position p (represented by $\mathbf{h}_p$), this layer uses the same scoring function as the pointer to compute the score of each possible label for that arc and assign the highest-scoring one:

$$\mathbf{u}_{tp}^{dl} = \mathbf{score}(\mathbf{s}_t^d, \mathbf{h}_p, l) = g_1(\mathbf{s}_t^d)^T W_l g_2(\mathbf{h}_p) + \mathbf{U}_l^T g_1(\mathbf{s}_t^d) + \mathbf{V}_l^T g_2(\mathbf{h}_p) + \mathbf{b}_l$$

where W_l, $\mathbf{U}_l$, $\mathbf{V}_l$ and $\mathbf{b}_l$ are parameters distinctly used for each label $l \in \{1, 2, \ldots, L\}$, being L the number of labels. In addition, $g_1(\cdot)$ and $g_2(\cdot)$ are two single-layer MLPs with ELU activation.

The described transition-based algorithm can produce unrestricted non-projective dependency structures in $O(n^2)$ time complexity, since each decoder d requires n attachments to successfully parse a sentence with n words, and at each step the attention vector $\mathbf{a}_t^d$ is computed over the whole input. Fig. 2 depicts a sketch of the multitask neural architecture and the decoding procedure for parsing the sentence in Fig. 1(a).

Multitask training. Following a multitask learning strategy [22], we jointly train a single neural model for more than one task by optimizing the sum of their objectives and sharing a common encoder representation.

As both tasks use a dependency representation, the training objective of the pointer of each decoder is to learn the probability $P_\theta(y|x)$, where y is the correct unlabeled dependency tree for a given sentence x: $P_\theta(y|x)$. This probability can be factorized to the sequence of SHIFT-ATTACH-p transitions to build y (this is basically the sequence of indices p_i):

$$P_\theta(y|x) = \prod_{i=1}^{n} P_\theta(p_i|p_{<i}, x)$$

where $p_{<i}$ represents previous predicted indices following the left-to-right order. We minimize the negative log of the probability of choosing the correct sequence of indices p implemented as cross-entropy loss:

$$\mathcal{L}_{pointer}^d = -\sum_{i=1}^{n} log P_\theta(p_i, p_{<i}, x)$$

Additionally, the labeler of each decoder is trained with softmax cross-entropy to minimize the negative log likelihood of tagging with the correct label l_i a given dependency arc defined between the head word in position p_i and the dependent word in the ith position:

$$\mathcal{L}_{labeler}^d = -\sum_{i=1}^{n} log P_\theta(l_i|p_i, i)$$

Then, the whole neural model is jointly trained by summing the pointer and labeler losses of each decoder:

$$\mathcal{L} = \mathcal{L}_{pointer}^{const} + \mathcal{L}_{labeler}^{const} + \mathcal{L}_{pointer}^{dep} + \mathcal{L}_{labeler}^{dep}$$

Finally, since both are considered main tasks and our goal is to train exclusively a single model, we neither use weights nor perform auxiliary-task training.

4. Experiments

4.1. Data

To test our approach, we focus on parallel data, where both constituent and dependency representations are available. In particular, we conduct experiments on well-known continuous datasets: the English Penn Treebank (PTB) [25] and its Stanford Dependencies [31] conversion (using the Stanford parser v3.3.0)[6] with standard splits; and the Chinese Penn Treebank 5.1 [26] and its converted dependency variant [35] with gold POS tags and two different splits: ZCTB [35], for dependency parsing, and LCTB [36], commonly used for constituent parsing. In addition, we undertake further experiments on two broadly-used discontinuous German treebanks and their available non-projective dependency representations: NEGRA [27] with standard splits [37] and TIGER [28] with the split provided in the SPMRL14 shared task [38,39]. For both datasets, we report results with and without gold POS tags.

For the constituent-to-dependency encoding, we identify head words on German constituents by applying the head-rule set defined by Rehbein [40] and, on English and Chinese structures, by using those developed by Collins [30] and Zhang and Clark [35], respectively. The resulting augmented dependencies match regular variants by around 70% in all languages, except for Chinese where the unlabeled augmented and regular dependency trees are exactly the same.

Following standard practice, we discard punctuation for evaluating on both Penn treebanks, using the EVALB script to report constituent accuracy. Furthermore, while all tokens are considered when reporting dependency performance on German datasets, we employ discodop[7] [41] and ignore punctuation and root symbols for evaluating on discontinuous constituent treebanks.

4.2. Settings

Word vectors are initialized with pre-trained structured-skipgram embeddings [42] for all languages and character and POS tag embeddings are randomly initialized. All of them are fine-tuned during training. POS tag embeddings are only enabled when gold information is used.

Additionally, we report accuracy gains by augmenting our model with the pre-trained language model BERT [34]. Although different approaches to initialize deep contextualized word embeddings from BERT can be found, we proceed with weights extracted from one or several layers for each token as a word-level representation. In addition, since BERT is trained on subwords, we take the vector of each subword of an input token w_i and use the average embedding as the final representation $\mathbf{e}_i^{BERT}$. In particular, we use in our experiments the pre-trained cased German and Chinese $\text{BERT}_{\text{BASE}}$ models with 12 768-dimensional hidden vectors; and uncased $\text{BERT}_{\text{LARGE}}$ with 24 1024-dimensional layers for English. Depending on the specific task, some layers proved to be more beneficial than others, which is especially crucial when the resulting embeddings are not fine-tuned during training. In order to check which layers are more suitable for our tasks, we test on development sets the combination of different layers. In Table 1, we compare, for the English pre-trained model $\text{BERT}_{\text{LARGE}}$, the accuracy obtained by averaging several groups of four consecutive layers (from last layer 24 to layer 13) and by just using weights from the second-to-last hidden layer (the simplest and commonly-used strategy, since it is less biased than the last layer to the target objectives used to train BERT). As can be seen, the combination of layers from 17 to 20 achieves the highest accuracy on both tasks and, therefore, this setup is used in our experiments on the PTB. Regarding the pre-trained models $\text{BERT}_{\text{BASE}}$ for German and Chinese, we noticed that comparable accuracies can be obtained by just using weights from the second-to-last layer instead of combining the four last layers as can be seen, for instance, in Table 2 for the NEGRA dataset. Therefore, we decided to follow the simplest configuration and use the second-to-last layer in all experiments on German and Chinese languages. We discarded other combinations such as the concatenation of several layers to avoid increasing the dimension of BERT embeddings. Finally, by adapting BERT-based embeddings to our specific tasks, our approach would certainly obtain some gains in accuracy; however, we consider that the amount of resources necessary to that end will not justify the expensive fine-tuning of parameter-heavy BERT layers.

Table 1
Accuracy comparison on regular and augmented dependency trees of the PTB development set by using weights from different BERT layers.

	Regular		Augmented	
	UAS	LAS	UAS	LAS
Layer 23	96.73	94.98	96.06	94.55
Layers 21–24	96.69	94.99	96.03	94.61
Layers 17–20	**96.88**	**95.13**	**96.19**	**94.75**
Layers 13–16	96.71	94.97	96.08	94.68

Table 2
Accuracy comparison on regular and augmented dependency trees of the NEGRA development set by using weights from different BERT layers.

	Regular		Augmented	
	UAS	LAS	UAS	LAS
Layer 11	**96.41**	95.56	**95.04**	94.48
Layers 9–12	96.40	**95.57**	95.02	**94.50**
Layers 5–8	96.31	95.50	94.89	94.40

In each training epoch, we use the same number of examples from each task and choose the multitask model with the highest harmonic mean among Labeled Attachment Scores on augmented and regular development sets. In addition, average accuracy over 3 repetitions is reported due to random initializations.

Finally, for parameter optimization and hyper-parameter selection, we follow Ma et al. [17] and Dozat and Manning [16] and these are detailed in Table 3. Please note that we use for the multitask variant the exact same hyper-parameters as the single-task baselines. By optimizing them to our specific multitask model, we could certainly increase performance; however, we decided to keep the same settings for a fair comparison.

4.3. Results

In Table 4, we compare our own implementation of the single-task dependency and constituent parsers by Fernández-González and Gómez-Rodríguez [18] and Fernández-González and Gómez-Rodríguez [21] to the proposed multitask approach. In all datasets tested, training a single model of the multi-representational parser across both syntactic representations leads to accuracy gains on both tasks.

In order to further put our approach into context, we also provide a comparison against state-of-the-art models. In Table 6, we show how our approach outperforms the best dependency parsers to date on the PTB and ZCTB with regular pre-trained word embeddings. Moreover, although some of the included parsers use several parameter-heavy layers of BERT and additionally perform a task-specific adaptation via expensive fine-tuning, our approach achieves similar performance on PTB and improves

[6] https://nlp.stanford.edu/software/lex-parser.shtml.
[7] https://github.com/andreasvc/disco-dop.

Table 3
Model hyper-parameters.

Architecture hyper-parameters	
BiLSTM encoder layers	3
BiLSTM encoder size	512
LSTM decoders layers	1
LSTM decoders size	512
LSTM layers dropout	0.33
CNN window size	3
CNN number of filters	50
Word/POS/Character embedding dimension	100
English BERT embedding dimension	1024
German BERT embedding dimension	768
Chinese BERT embedding dimension	768
Embeddings dropout	0.33
MLP layers	1
MLP activation function	ELU
Arc MLP size	512
Label MLP size	128
UNK replacement probability	0.5
Beam size	10
Optimizer	Adam [43]
Initial learning rate	0.001
β_1, β_2	0.9
Batch size	32
Decay rate	0.75
Gradient clipping	5.0

Table 4
Accuracy comparison of single-task baseline parsers to the proposed multi-representational approach in both constituent and dependency parsing. We report Labeled Attachment Scores (LAS) and Unlabeled Attachment Scores (UAS) for dependency parsing and, for constituent parsing, the LAS on the augmented dependency trees and F-score on the post-decoding constituent structure. The corresponding standard deviations over 3 runs for each score are reported in Table 5.

Treebank	Single-dep.		Single-const.	Multi-representational		
	UAS	LAS	F1 (LAS)	UAS	LAS	F1 (LAS)
PTB_{noPOS}	96.06	94.50	93.29 (93.57)	**96.25**	**94.64**	**93.67 (93.93)**
$LCTB_{gold}$	93.26	92.67	88.28 (88.49)	**93.40**	**92.88**	**88.65 (88.61)**
$ZCTB_{gold}$	90.61	89.51	86.01 (84.38)	**90.79**	**89.69**	**86.09 (84.43)**
$NEGRA_{gold}$	94.71	93.87	86.42 (92.22)	**94.80**	**94.05**	**87.30 (92.68)**
$NEGRA_{noPOS}$	94.20	93.19	85.65 (91.36)	**94.33**	**93.33**	**86.78 (91.85)**
$TIGER_{gold}$	94.24	92.86	86.74 (91.81)	**94.31**	**92.90**	**87.25 (92.22)**
$TIGER_{noPOS}$	93.73	92.27	85.96 (90.89)	**93.85**	**92.35**	**86.61 (91.36)**

Table 5
Standard deviations of scores in Table 4 over 3 runs.

Treebank	Single-Dep.		Single-Const.	Multi-Representational		
	UAS	LAS	F1 (LAS)	UAS	LAS	F1 (LAS)
PTB_{noPOS}	±0.03	±0.04	±0.06 (±0.04)	±0.04	±0.04	±0.05 (±0.03)
$LCTB_{gold}$	±0.08	±0.09	±0.06 (±0.04)	±0.07	±0.08	±0.09 (±0.07)
$ZCTB_{gold}$	±0.07	±0.05	±0.07 (±0.06)	±0.08	±0.06	±0.07 (±0.05)
$NEGRA_{gold}$	±0.03	±0.06	±0.06 (±0.04)	±0.02	±0.03	±0.04 (±0.02)
$NEGRA_{noPOS}$	±0.04	±0.04	±0.05 (±0.03)	±0.06	±0.04	±0.06 (±0.03)
$TIGER_{gold}$	±0.04	±0.05	±0.06 (±0.04)	±0.03	±0.05	±0.04 (±0.02)
$TIGER_{noPOS}$	±0.07	±0.05	±0.06 (±0.06)	±0.05	±0.04	±0.07 (±0.05)

over all models on ZCTB. We also outperform the single-task dependency parser by Fernández-González and Gómez-Rodríguez [18] with BERT, providing evidence that our multitask neural architecture is learning extra syntactic information that is not encoded in the pre-trained model BERT. Furthermore, Table 7 shows that our novel parser obtains competitive accuracies on constituent PTB and LCTB without BERT (best F-score to date on the latter), while being more efficient than $O(n^3)$ and $O(n^5)$ approaches such as [14,44]. Finally, in Table 8 we show how our novel neural architecture outperforms all existing single-task parsers on the discontinuous NEGRA and TIGER datasets with regular word embeddings.

Table 6
Accuracy comparison of state-of-the-art dependency parsers on PTB and ZCTB. Since in the original work [18] performance with BERT was not reported, we run our own implementation of the single-task dependency parser enhanced with BERT-based embeddings and include it in the second block as "Fernández-González and Gómez-Rodríguez [18]".

Parser	PTB		ZCTB	
	UAS	LAS	UAS	LAS
Wang and Chang [45]	94.08	91.82	87.55	86.23
Cheng et al. [46]	94.10	91.49	88.1	85.7
Kuncoro et al. [47]	94.26	92.06	88.87	87.30
Zhang et al. [48]	93.42	91.29	87.65	86.17
Zhang et al. [49]	94.10	91.90	87.84	86.15
Ma and Hovy [50]	94.88	92.96	89.05	87.74
Dozat and Manning [16]	95.74	94.08	89.30	88.23
Li et al. [51]	94.11	92.08	88.78	86.23
Ma et al. [17]	95.87	94.19	90.59	89.29
Ji et al. [52]	95.97	94.31	–	–
Fernández-González and Gómez-Rodríguez [18]	96.04	94.43	–	–
Zhou and Zhao [14]	96.09	**94.68**	–	–
Li et al. [53]	95.83	94.54	90.47	89.44
Zhang et al. [54]	96.14	94.49	–	–
This work	**96.25**	94.64	**90.79**	**89.69**
+BERT				
Fernández-González and Gómez-Rodríguez [18]	96.91	95.35	92.58	91.42
Li et al. [53]	96.44	94.63	90.89	89.73
Li et al. [53][a]	96.57	95.05	–	–
Zhou and Zhao [14][a]	**97.00**	95.43	91.21	89.15
This work	96.97	**95.46**	**92.78**	**91.65**

[a]Models that fine-tune BERT.

Table 7
F-score comparison of state-of-the-art constituent parsers on PTB and LCTB.

Parser	PTB	LCTB
Dyer et al. [55]	91.2	84.6
Cross and Huang [56]	91.3	–
Liu and Zhang [36]	91.7	85.5
Liu and Zhang [57]	91.8	86.1
Fernández-González and Gómez-Rodríguez [58]	92.0	86.6
Stern et al. [59]	91.8	–
Stern et al. [60]	92.56	–
Shen et al. [61]	–	86.5
Fried and Klein [62]	92.2	87.0
Gaddy et al. [63]	92.08	–
Teng and Zhang [64]	92.4	87.3
Kitaev and Klein [44]	93.55	–
Zhou and Zhao [14]	**93.78**	–
This work	93.67	**88.65**
+BERT		
Kitaev et al. [65][a]	95.59	91.75
Zhou and Zhao [14][a]	**95.84**	**92.18**
This work	95.23	90.20

[a]Models that fine-tune BERT.

4.4. Analysis

In order to obtain insight into why the multi-representational variant is outperforming single-task parsers in both tasks,[8] we conduct an error analysis relative to structural factors.

For the dependency parsing task, we show in Fig. 3(a) the F-score relative to dependency displacements (i.e., signed distances) on the PTB and on the concatenation of all datasets,[9] Fig. 3(b) reports the performance on common dependency relations on PTB and Fig. 3(c) shows the accuracy of both approaches relative to sentence lengths on PTB and on all datasets together.

[8] Apart from the widely-proven benefits of using multitask learning as a regularization method to avoid overfitting.

[9] We discard German datasets with gold PoS tags.

From these results, we can point out that the multitask parser is performing better on longer leftward dependency arcs (with positive displacement) and on longer sentences, improving over the single-task system in all frequent dependency relations.

Regarding constituent parsing, we specifically analyze performance on both discontinuous German datasets together, where the multi-representational model significantly outperforms the single-task approach. Firstly, we report in Table 8 an F-score exclusively measured on discontinuous constituents (DF1), showing a notable performance on discontinuous structures (probably thanks to the joint training with regular non-projective dependency structures). Additionally, Fig. 3(d) plots the F-score on span identification for different lengths, Fig. 3(e) shows the performance by span labels and Fig. 3(f) measures the accuracy of both approaches on different sentence length cutoffs. It can be noticed that the multitask variant achieves higher performance when spans are larger and sentences tend to be longer, being only less accurate than the single-task parser on Coordinated Noun Phrases (CNP), where, in this particular case, a disagreement in notation between constituent and dependency representations[10] might be misleading the multitask approach.

All this provides some evidences that learning across syntactic representations is tackling the main weakness of the transition-based sequential decoding: the impact of error propagation on the performance on large constituents and long sentences. Moreover, the information exclusively encoded by each formalism (span phrase information in constituent trees and semantic relations in dependency structures) may complete each other and provide an additional guidance not only in final decoding steps (where the parser is more prone to make a mistake due to error propagation), but also in creating those structures that are less frequent in some of the two representations (as happens with long leftward dependency arcs in languages such as English).

It is also worth mentioning that even on Chinese datasets (where augmented and regular dependencies are the same) our approach benefits from learning across both structures, meaning that both constituent-based and regular dependency label sets provide useful syntactic information.

Finally, the multitask approach achieves lower accuracies on continuous constituent datasets since the encoding technique by Fernández-González and Martins [23] cannot directly handle unary nodes (which are collapsed or, in case of leaf unary nodes, assigned with a regular sequence tagger), losing some accuracy in continuous treebanks where the amount of this kind of structures is significant: 19.69% and 19.09% of the constituents on the PTB training and development sets, respectively, are unary nodes. One consequence of encoding unaries by collapsing them is that, while the labeler on regular dependency trees deals with 47 different dependency labels on the PTB, the labeler on augmented dependency structures manages 188 different tags (104 of them being generated for encoding unary nodes). On the contrary, in discontinuous datasets such as TIGER (where unary nodes are discarded due to their low frequency), the regular label set size is 45 and the augmented version has 83. This significant increase on augmented dictionary sizes for processing continuous datasets might penalize the labeler's performance and affect final accuracy, especially in an encoding technique where dependency labels have a crucial role during constituent recovery. Additionally, the recovery of leaf unary nodes (the 73.55% of total unaries from PTB development set for example) lost after the constituent-to-dependency conversion has a greater impact on final accuracy.

[10] In the regular dependency version, a CNP structure is represented by attaching the second noun to the conjunction and the latter to the first noun, while in the augmented variant, the first noun and the conjunction are both attached to the second noun.

Table 8
F-score and Discontinuous F-score (DF1) comparison of state-of-the-art discontinuous constituent parsers on NEGRA and TIGER.

Parser	NEGRA		TIGER	
	F1	DF1	F1	DF1
(Predicted/Without PoS tags)				
Fernández-González and Martins [23]	77.0	–	77.3	–
Versley [66]	–	–	79.5	–
Stanojević and G. Alhama [67]	–	–	77.0	–
Coavoux and Crabbé [68]	–	–	79.3	–
Coavoux et al. [20]	83.2	54.6	82.7	55.9
Coavoux and Cohen [19]	83.2	56.3	82.5	55.9
Stanojević and Steedman [69]	83.6	50.7	83.4	53.5
Vilares and Gómez-Rodríguez [70]	75.6	34.6	77.5	39.5
Fernández-González and Gómez-Rodríguez [21]	85.7	58.6	85.7	60.4
Corro [71]	86.3	56.1	85.2	51.2
This work	**86.8**	**69.5**	**86.6**	**62.6**
+BERT				
Vilares and Gómez-Rodríguez [70][a]	83.9	45.6	84.6	51.1
Corro [71][a]	**91.6**	66.1	**90.0**	62.1
Fernández-González and Gómez-Rodríguez [72][a]	90.4	66.5	88.5	62.7
This work	91.0	**76.6**	89.8	**71.0**
(Gold PoS tags)				
Maier [73]	77.0	19.8	74.7	18.8
Fernández-González and Martins [23]	80.5	–	80.6	–
Maier and Lichte [74]	–	–	76.5	–
Corro et al. [75]	–	–	81.6	–
Stanojević and G. Alhama [67]	82.9	–	81.6	–
Coavoux and Crabbé [68]	82.2	50.0	81.6	49.2
Gebhardt [76]	–	–	75.1	–
Mörbitz and Ruprecht [77]	82.8	52.9	81.8	54.6
Vilares and Gómez-Rodríguez [70]	77.1	36.5	79.2	40.1
Fernández-González and Gómez-Rodríguez [21]	86.1	59.9	86.3	60.7
This work	**87.3**	**71.0**	**87.3**	**64.2**

[a] Models that fine-tune BERT.

The tagger in charge of that has to face a complex task, since the amount of words with unary constituents on top is scarce on the training set (88.85% of words are tagged with NONE and, since a sequence of leaf unaries is collapsed into a single tag as done for non-leaf unary nodes, the model has to deal with a large dictionary size of 54 tags), hindering the adequate training of the tagger. While it achieves a good overall accuracy (for instance, 98.65% on the PTB development set), a worse performance is obtained when only considering words with attached unary nodes (just the 10.59% of total words): 92.56% recall, 91.82% precision and 92.19% F-score on the PTB development set. It might seem that this performance is good enough; however, it means that tagging errors are more than 5 times as frequent in words associated with unary nodes compared to the overall error rate, and its impact on the final parsing accuracy is significant taking into account that scores on Penn treebanks are remarkably high. Despite all that, our approach obtains the best accuracy to date among all existing transition-based parsers in both continuous and discontinuous constituent structures, and it is on par with state-of-the-art models such as [44] and [14].

5. Related work

It is known that parsers based on lexicalized grammar are trained using both constituent and unlabeled dependency information. This includes classic chart parsers [78] as well as lexicalized parsers that build dependencies with reduce transitions, such as [79], which can generate both structures. These are restricted to dependencies that are directly inferred from the lexicalized constituent trees. In this sense, the multitask approach is more flexible, as it does not have that limitation and one can use dependencies and constituents from different sources.

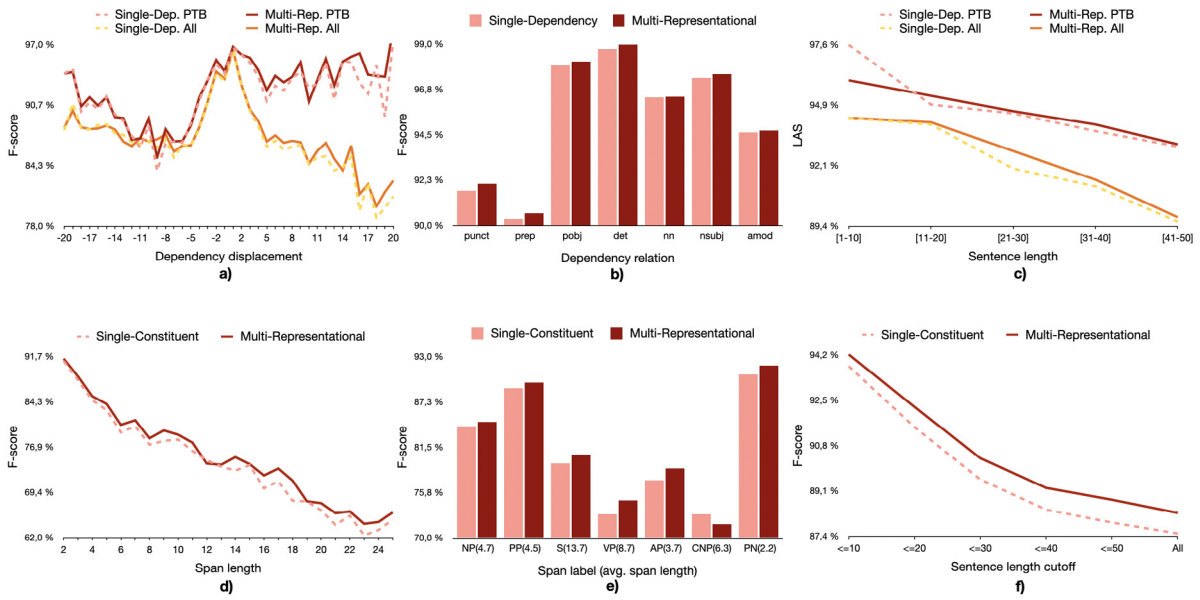

Fig. 3. Parsing performance of the single-task and the multi-representational parsers relative to length and structural factors.

In the deep learning era, there have been a few recent attempts to jointly train a neural model across constituent and dependency trees, producing, during decoding, both syntactic representations from a single model.

In particular, Strzyz et al. [15] propose a multitask sequence labeling architecture that, by representing constituent and dependency trees as linearizations [80,81], can learn and perform parsing in both formalisms as joint tasks. While being a linear and fast parser, the parsing accuracy provided by this approach is notably behind the state of the art (even training separate models by performing an auxiliary-task learning for each formalism) and the linearization strategy used for constituent parsing is restricted to continuous structures.

Zhou and Zhao [14] also explore the benefits of training a model across syntactic representations. They propose to integrate dependency and constituent information into a simplified variant of the Head-Driven Phrase Structure Grammar formalism (HPSG). Then, to implement a HPSG parser, they modify the constituent chart-based parser by Kitaev and Klein [44] that employs a $O(n^5)$ CKY-style algorithm [60] for decoding.[11] Although their approach can produce both syntactic structures at the same time and achieve state-of-the-art accuracies on PTB and CTB treebanks, their parser is bounded to produce continuous and projective structures with a high runtime complexity.

Our approach can handle any kind of constituent and dependency structures and provides an efficient runtime complexity, crucial for some downstream applications.

6. Conclusions and future work

We propose a novel encoder–decoder neural architecture based on Pointer Networks that, after being jointly trained on regular and constituent-based dependency trees, can syntactically parse a sentence to both constituent and dependency trees. Apart from just requiring to train a single model, our approach can produce not only the simplest continuous/projective trees, but also discontinuous/non-projective structures in just $O(n^2)$ runtime. We test our parser on the main dependency and constituent benchmarks, obtaining competitive results in all cases and reporting state-of-the-art accuracies in several datasets.

As future work, we plan to perform auxiliary-task learning and train a separate model for each task, testing different weights for the loss computation. This will lose the advantage of training a single model to undertake both tasks, but will certainly lead to further improvements in accuracy.

CRediT authorship contribution statement

Daniel Fernández-González: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Carlos Gómez-Rodríguez:** Conceptualization, Validation, Formal analysis, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We acknowledge the European Research Council (ERC), which has funded this research under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), ERDF/MICINN-AEI (ANSWER-ASAP, TIN2017-85160-C2-1-R; SCANNER-UDC, PID2020-113230RB-C21), Xunta de Galicia, Spain (ED431C 2020/11), and Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia, Spain and the European Union (ERDF - Galicia 2014–2020 Program), by grant ED431G 2019/01. Funding for open access charge: Universidade da Coruña/ CISUG.

References

[1] M. Zhang, Z. Li, G. Fu, M. Zhang, Syntax-enhanced neural machine translation with syntax-aware word representations, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 1151–1161, http://dx.doi.org/10.18653/v1/N19-1118, URL: https://www.aclweb.org/anthology/N19-1118.

[11] They also propose a $O(n^3)$ decoding method that achieves worse accuracy.

[2] B. Yang, D.F. Wong, L.S. Chao, M. Zhang, Improving tree-based neural machine translation with dynamic lexicalized dependency encoding, Knowl.-Based Syst. 188 (2020) 105042, http://dx.doi.org/10.1016/j.knosys.2019.105042, URL: https://www.sciencedirect.com/science/article/pii/S095070511930440X.

[3] M. Zhang, Z. Li, G. Fu, M. Zhang, Dependency-based syntax-aware word representations, Artificial Intelligence 292 (2021) 103427, http://dx.doi.org/10.1016/j.artint.2020.103427, URL: https://www.sciencedirect.com/science/article/pii/S0004370220301764.

[4] D.Q. Nguyen, K. Verspoor, From POS tagging to dependency parsing for biomedical event extraction, BMC Bioinformatics 20 (1) (2019) 72, http://dx.doi.org/10.1186/s12859-019-2604-0.

[5] B. Zhang, Y. Zhang, R. Wang, Z. Li, M. Zhang, Syntax-aware opinion role labeling with dependency graph convolutional networks, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 3249–3258, http://dx.doi.org/10.18653/v1/2020.acl-main.297, URL: https://www.aclweb.org/anthology/2020.acl-main.297.

[6] Q. Xia, B. Zhang, R. Wang, Z. Li, Y. Zhang, F. Huang, L. Si, M. Zhang, A unified span-based approach for opinion mining with syntactic constituents, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 1795–1804, URL: https://www.aclweb.org/anthology/2021.naacl-main.144.

[7] Q. Cao, X. Liang, B. Li, L. Lin, Interpretable visual question answering by reasoning on dependency trees, IEEE Trans. Pattern Anal. Mach. Intell. 43 (3) (2021) 887—901, http://dx.doi.org/10.1109/tpami.2019.2943456.

[8] Z. Xu, D. Guo, D. Tang, Q. Su, L. Shou, M. Gong, W. Zhong, X. Quan, N. Duan, D. Jiang, Syntax-enhanced pre-trained model, 2021, arXiv:2012.14116.

[9] J. Bai, Y. Wang, Y. Chen, Y. Yang, J. Bai, J. Yu, Y. Tong, Syntax-BERT: Improving pre-trained transformers with syntax trees, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 3011–3020, URL: https://www.aclweb.org/anthology/2021.eacl-main.262.

[10] V. Balachandran, A. Pagnoni, J.Y. Lee, D. Rajagopal, J. Carbonell, Y. Tsvetkov, StructSum: Summarization via structured representations, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 2575–2585, URL: https://www.aclweb.org/anthology/2021.eacl-main.220.

[11] D. Sachan, Y. Zhang, P. Qi, W.L. Hamilton, Do syntax trees help pre-trained transformers extract information? in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 2647–2661, URL: https://www.aclweb.org/anthology/2021.eacl-main.228.

[12] S. Müller, Continuous or discontinuous constituents? A comparison between syntactic analyses for constituent order and their processing systems, Res. Lang. Comput. 2 (2) (2004) 209–257.

[13] S. Kahane, N. Mazziotta, Syntactic polygraphs. A formalism extending both constituency and dependency, in: Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015), Association for Computational Linguistics, Chicago, USA, 2015, pp. 152–164, URL: http://www.aclweb.org/anthology/W15-2313.

[14] J. Zhou, H. Zhao, Head-driven phrase structure grammar parsing on penn treebank, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2396–2408, http://dx.doi.org/10.18653/v1/P19-1230, URL: https://www.aclweb.org/anthology/P19-1230.

[15] M. Strzyz, D. Vilares, C. Gómez-Rodríguez, Sequence labeling parsing by learning across representations, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 5350–5357, http://dx.doi.org/10.18653/v1/P19-1531, URL: https://www.aclweb.org/anthology/P19-1531.

[16] T. Dozat, C.D. Manning, Deep biaffine attention for neural dependency parsing, in: ICLR, OpenReview.net, 2017.

[17] X. Ma, Z. Hu, J. Liu, N. Peng, G. Neubig, E.H. Hovy, Stack-pointer networks for dependency parsing, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, July 15-20, 2018, 2018, pp. 1403–1414, URL: https://aclanthology.info/papers/P18-1130/p18-1130.

[18] D. Fernández-González, C. Gómez-Rodríguez, Left-to-right dependency parsing with pointer networks, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 710–716, http://dx.doi.org/10.18653/v1/N19-1076, URL: https://www.aclweb.org/anthology/N19-1076.

[19] M. Coavoux, S.B. Cohen, Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 204–217, URL: https://www.aclweb.org/anthology/N19-1018.

[20] M. Coavoux, B. Crabbé, S.B. Cohen, Unlexicalized transition-based discontinuous constituency parsing, Trans. Assoc. Comput. Linguist. 7 (2019) 73–89, http://dx.doi.org/10.1162/tacl_a_00255, URL: https://www.aclweb.org/anthology/Q19-1005.

[21] D. Fernández-González, C. Gómez-Rodríguez, Discontinuous constituent parsing with pointer networks, in: Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 7724–7731, http://dx.doi.org/10.1609/aaai.v34i05.6275, URL: https://aaai.org/ojs/index.php/AAAI/article/view/6275.

[22] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75, http://dx.doi.org/10.1023/A:1007379606734.

[23] D. Fernández-González, A.F.T. Martins, Parsing as reduction, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics), Association for Computational Linguistics, Beijing, China, 2015, pp. 1523–1533, http://dx.doi.org/10.3115/v1/P15-1147, URL: https://www.aclweb.org/anthology/P15-1147.

[24] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, in: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2692–2700, URL: http://papers.nips.cc/paper/5866-pointer-networks.pdf.

[25] M.P. Marcus, B. Santorini, M.A. Marcinkiewicz, Building a large annotated corpus of english: The penn treebank, Comput. Linguist. 19 (1993) 313–330.

[26] N. Xue, F. Xia, F.-d. Chiou, M. Palmer, The penn Chinese TreeBank: Phrase structure annotation of a large corpus, Nat. Lang. Eng. 11 (2) (2005) 207–238, http://dx.doi.org/10.1017/S135132490400364X.

[27] W. Skut, B. Krenn, T. Brants, H. Uszkoreit, An annotation scheme for free word order languages, in: Proceedings of the Fifth Conference on Applied Natural Language Processing, in: ANLC '97, Association for Computational Linguistics, Stroudsburg, PA, USA, 1997, pp. 88–95, http://dx.doi.org/10.3115/974557.974571.

[28] S. Brants, S. Dipper, S. Hansen, W. Lezius, G. Smith, TIGER treebank, in: Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT), 2002, pp. 24–42.

[29] J. Yang, Y. Zhang, NCRF++: An open-source neural sequence labeling toolkit, in: Proceedings of ACL 2018, System Demonstrations, Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 74–79, http://dx.doi.org/10.18653/v1/P18-4013, URL: https://www.aclweb.org/anthology/P18-4013.

[30] M. Collins, Head-Driven Statistical Models for Natural Language Parsing (Ph.D. thesis), University of Pennsylvania, 1999.

[31] M.-C. de Marneffe, C.D. Manning, The stanford typed dependencies representation, in: Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, Coling 2008 Organizing Committee, Manchester, UK, 2008, pp. 1–8, URL: https://www.aclweb.org/anthology/W08-1301.

[32] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, CoRR abs/1409.0473.

[33] X. Ma, E. Hovy, End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2016, pp. 1064–1074, http://dx.doi.org/10.18653/v1/P16-1101, URL: http://aclweb.org/anthology/P16-1101.

[34] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186, http://dx.doi.org/10.18653/v1/N19-1423, URL: https://www.aclweb.org/anthology/N19-1423.

[35] Y. Zhang, S. Clark, A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing, in: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Honolulu, Hawaii, 2008, pp. 562–571, URL: https://www.aclweb.org/anthology/D08-1059.

[36] J. Liu, Y. Zhang, Shift-reduce constituent parsing with neural lookahead features, Trans. Assoc. Comput. Linguist. 5 (2017) 45–58, http://dx.doi.org/10.1162/tacl_a_00045, URL: https://www.aclweb.org/anthology/Q17-1004.

[37] A. Dubey, F. Keller, Probabilistic parsing for german using sister-head dependencies, in: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Sapporo, Japan, 2003, pp. 96–103, http://dx.doi.org/10.3115/1075096.1075109, URL: https://www.aclweb.org/anthology/P03-1013.

[38] D. Seddah, R. Tsarfaty, S. Kübler, M. Candito, J.D. Choi, R. Farkas, J. Foster, I. Goenaga, K. Gojenola Galletebeitia, Y. Goldberg, S. Green, N. Habash, M. Kuhlmann, W. Maier, J. Nivre, A. Przepiórkowski, R. Roth, W. Seeker, Y. Versley, V. Vincze, M. Woliński, A. Wróblewska, E. Villemonte de la Clergerie, Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages, in: Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages, Association for Computational Linguistics, Seattle, Washington, USA, 2013, pp. 146–182, URL: https://www.aclweb.org/anthology/W13-4917.

[39] B. Crabbé, Multilingual discriminative shift-reduce phrase structure parsing for the SPMRL 2014 shared task, 2014.

[40] I. Rehbein, Treebank-Based Grammar Acquisition for German (Ph.D. thesis), Dublin City University, Dublin, 2009, p. 249, URL: http://nbn-resolving.de/urn:nbn:de:hebis:30:3-330238.

[41] A. van Cranenburgh, R. Scha, R. Bod, Data-oriented parsing with discontinuous constituents and function tags, J. Lang. Model. 4 (2016) 57–111.

[42] W. Ling, C. Dyer, A.W. Black, I. Trancoso, Two/too simple adaptations of Word2Vec for syntax problems, in: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Denver, Colorado, 2015, pp. 1299–1304, http://dx.doi.org/10.3115/v1/N15-1142, URL: https://www.aclweb.org/anthology/N15-1142.

[43] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, URL: http://arxiv.org/abs/1412.6980. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[44] N. Kitaev, D. Klein, Constituency parsing with a self-attentive encoder, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 2676–2686, http://dx.doi.org/10.18653/v1/P18-1249, URL: https://www.aclweb.org/anthology/P18-1249.

[45] W. Wang, B. Chang, Graph-based dependency parsing with bidirectional LSTM, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2016, pp. 2306–2315, http://dx.doi.org/10.18653/v1/P16-1218, URL: http://aclweb.org/anthology/P16-1218.

[46] H. Cheng, H. Fang, X. He, J. Gao, L. Deng, Bi-directional attention with agreement for dependency parsing, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2016, pp. 2204–2214, http://dx.doi.org/10.18653/v1/D16-1238, URL: http://aclweb.org/anthology/D16-1238.

[47] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, N.A. Smith, Distilling an ensemble of greedy dependency parsers into one MST parser, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2016, pp. 1744–1753, http://dx.doi.org/10.18653/v1/D16-1180, URL: http://aclweb.org/anthology/D16-1180.

[48] Z. Zhang, H. Zhao, L. Qin, Probabilistic graph-based dependency parsing with convolutional neural network, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1382–1392, http://dx.doi.org/10.18653/v1/P16-1131, URL: https://www.aclweb.org/anthology/P16-1131.

[49] X. Zhang, J. Cheng, M. Lapata, Dependency parsing as head selection, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers, 2017, pp. 665–676, URL: https://aclanthology.info/papers/E17-1063/e17-1063.

[50] X. Ma, E. Hovy, Neural probabilistic model for non-projective MST parsing, in: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Asian Federation of Natural Language Processing, 2017, pp. 59–69, URL: http://aclweb.org/anthology/I17-1007.

[51] Z. Li, J. Cai, S. He, H. Zhao, Seq2seq dependency parsing, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 3203–3214, URL: https://www.aclweb.org/anthology/C18-1271.

[52] T. Ji, Y. Wu, M. Lan, Graph-based dependency parsing with graph neural networks, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2475–2485, http://dx.doi.org/10.18653/v1/P19-1237, URL: https://www.aclweb.org/anthology/P19-1237.

[53] Z. Li, H. Zhao, K. Parnow, Global greedy dependency parsing, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2020), 2020.

[54] Y. Zhang, Z. Li, M. Zhang, Efficient second-order treeCRF for neural dependency parsing, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 3295–3305, http://dx.doi.org/10.18653/v1/2020.acl-main.302, URL: https://www.aclweb.org/anthology/2020.acl-main.302.

[55] C. Dyer, A. Kuncoro, M. Ballesteros, N.A. Smith, Recurrent neural network grammars, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, San Diego, California, 2016, pp. 199–209, http://dx.doi.org/10.18653/v1/N16-1024, URL: https://www.aclweb.org/anthology/N16-1024.

[56] J. Cross, L. Huang, Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 1–11, http://dx.doi.org/10.18653/v1/D16-1001, URL: https://www.aclweb.org/anthology/D16-1001.

[57] J. Liu, Y. Zhang, In-order transition-based constituent parsing, Trans. Assoc. Comput. Linguist. 5 (2017) 413–424, http://dx.doi.org/10.1162/tacl_a_00070, URL: https://www.aclweb.org/anthology/Q17-1029.

[58] D. Fernández-González, C. Gómez-Rodríguez, Dynamic oracles for top-down and in-order shift-reduce constituent parsing, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 1303–1313, http://dx.doi.org/10.18653/v1/D18-1161, URL: https://www.aclweb.org/anthology/D18-1161.

[59] M. Stern, J. Andreas, D. Klein, A minimal span-based neural constituency parser, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 818–827, http://dx.doi.org/10.18653/v1/P17-1076, URL: https://www.aclweb.org/anthology/P17-1076.

[60] M. Stern, D. Fried, D. Klein, Effective inference for generative neural parsing, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 1695–1700, http://dx.doi.org/10.18653/v1/D17-1178, URL: https://www.aclweb.org/anthology/D17-1178.

[61] Y. Shen, Z. Lin, A.P. Jacob, A. Sordoni, A. Courville, Y. Bengio, Straight to the tree: Constituency parsing with neural syntactic distance, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 1171–1180, http://dx.doi.org/10.18653/v1/P18-1108, URL: https://www.aclweb.org/anthology/P18-1108.

[62] D. Fried, D. Klein, Policy gradient as a proxy for dynamic oracles in constituency parsing, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 469–476, http://dx.doi.org/10.18653/v1/P18-2075, URL: https://www.aclweb.org/anthology/P18-2075.

[63] D. Gaddy, M. Stern, D. Klein, What's going on in neural constituency parsers? An analysis, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 999–1010, http://dx.doi.org/10.18653/v1/N18-1091, URL: https://www.aclweb.org/anthology/N18-1091.

[64] Z. Teng, Y. Zhang, Two local models for neural constituent parsing, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 119–132, URL: https://www.aclweb.org/anthology/C18-1011.

[65] N. Kitaev, S. Cao, D. Klein, Multilingual constituency parsing with self-attention and pre-training, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 3499–3505, http://dx.doi.org/10.18653/v1/P19-1340, URL: https://www.aclweb.org/anthology/P19-1340.

[66] Y. Versley, Discontinuity (Re)^2-visited: A minimalist approach to pseudoprojective constituent parsing, in: Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing, Association for Computational Linguistics, San Diego, California, 2016, pp. 58–69, http://dx.doi.org/10.18653/v1/W16-0907, URL: https://www.aclweb.org/anthology/W16-0907.

[67] M. Stanojević, R. G. Alhama, Neural discontinuous constituency parsing, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 1666–1676, http://dx.doi.org/10.18653/v1/D17-1174, URL: https://www.aclweb.org/anthology/D17-1174.

[68] M. Coavoux, B. Crabbé, Incremental discontinuous phrase structure parsing with the GAP transition, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 1259–1270, URL: https://www.aclweb.org/anthology/E17-1118.

[69] M. Stanojević, M. Steedman, Span-based LCFRS-2 parsing, in: Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing Into Enhanced Universal Dependencies, Association for Computational Linguistics, Online, 2020, pp.

111–121, http://dx.doi.org/10.18653/v1/2020.iwpt-1.12, URL: https://www.aclweb.org/anthology/2020.iwpt-1.12.
[70] D. Vilares, C. Gómez-Rodríguez, Discontinuous constituent parsing as sequence labeling, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 2771–2785, http://dx.doi.org/10.18653/v1/2020.emnlp-main.221, URL: https://www.aclweb.org/anthology/2020.emnlp-main.221.
[71] C. Corro, Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from O(nˆ6) down to O(nˆ3), in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 2753–2764, http://dx.doi.org/10.18653/v1/2020.emnlp-main.219, URL: https://www.aclweb.org/anthology/2020.emnlp-main.219.
[72] D. Fernández-González, C. Gómez-Rodríguez, Reducing discontinuous to continuous parsing with pointer network reordering, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 10570–10578, https://aclanthology.org/2021.emnlp-main.825.
[73] W. Maier, Discontinuous incremental shift-reduce parsing, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Beijing, China, 2015, pp. 1202–1212, http://dx.doi.org/10.3115/v1/P15-1116, URL: https://www.aclweb.org/anthology/P15-1116.
[74] W. Maier, T. Lichte, Discontinuous parsing with continuous trees, in: Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing, Association for Computational Linguistics, San Diego, California, 2016, pp. 47–57, http://dx.doi.org/10.18653/v1/W16-0906, URL: https://www.aclweb.org/anthology/W16-0906.
[75] C. Corro, J. Le Roux, M. Lacroix, Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 1644–1654, http://dx.doi.org/10.18653/v1/D17-1172, URL: https://www.aclweb.org/anthology/D17-1172.
[76] K. Gebhardt, Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 3049–3063, URL: https://www.aclweb.org/anthology/C18-1258.
[77] R. Mörbitz, T. Ruprecht, Supertagging-based parsing with linear context-free rewriting systems, 2020, arXiv:2010.10238.
[78] M. Collins, Head-driven statistical models for natural language parsing, Comput. Linguist. 29 (4) (2003) 589–637, http://dx.doi.org/10.1162/089120103322753356, URL: https://www.aclweb.org/anthology/J03-4003.
[79] B. Crabbé, Multilingual discriminative lexicalized phrase structure parsing, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1847–1856, URL: http://aclweb.org/anthology/D15-1212.
[80] C. Gómez-Rodríguez, D. Vilares, Constituent parsing as sequence labeling, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 1314–1324, http://dx.doi.org/10.18653/v1/D18-1162, URL: https://www.aclweb.org/anthology/D18-1162.
[81] M. Strzyz, D. Vilares, C. Gómez-Rodríguez, Viable dependency parsing as sequence labeling, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 717–723, http://dx.doi.org/10.18653/v1/N19-1077, URL: https://www.aclweb.org/anthology/N19-1077.

Daniel Fernández-González

Carlos Gómez-Rodríguez