Dependency Graph Parsing as Sequence Labeling

Ana Ezquerro, David Vilares, Carlos Gómez-Rodríguez
Universidade da Coruña, CITIC
Departamento de Ciencias de la Computación y Tecnologías de la Información
Campus de Elviña s/n, 15071
A Coruña, Spain
{ana.ezquerro, david.vilares, carlos.gomez}@udc.es

Abstract

Various linearizations have been proposed to cast syntactic dependency parsing as sequence labeling. However, these approaches do not support more complex graph-based representations, such as semantic dependencies or enhanced universal dependencies, as they cannot handle reentrancy or cycles. By extending them, we define a range of unbounded and bounded linearizations that can be used to cast graph parsing as a tagging task, enlarging the toolbox of problems that can be solved under this paradigm. Experimental results on semantic dependency and enhanced UD parsing show that with a good choice of encoding, sequence-labeling dependency graph parsers combine high efficiency with accuracies close to the state of the art, in spite of their simplicity.

1 Introduction

In recent years, a new family of approaches has emerged for dependency parsing that treats the problem as a sequence labeling task (Strzyz et al., 2019). This has advantages in terms of simplicity, flexibility and speed, as parsing can be performed with generic sequence labeling software and easily combined with other tasks that operate within the same framework. For this, one needs an encoding that can represent parse trees as a sequence composed of one discrete label per word, so that a sequence labeling component can be trained and output trees can then be decoded from the sequences.

In the last few years, a wide range of such encodings with different strengths and weaknesses have been proposed for dependency parsing (Strzyz et al., 2019; Lacroix, 2019; Strzyz et al., 2020; Gómez-Rodríguez et al., 2023; Amini et al., 2023). However, these encodings are designed for dependency trees, where each word is restricted to have exactly one parent and cycles are forbidden. The more complex family of structured prediction problems where the output is a graph of dependencies between words, including tasks like semantic dependency parsing (Oepen et al., 2015), enhanced Universal Dependencies parsing (Bouma et al., 2021) or even structured sentiment analysis (Barnes et al., 2021); has not been integrated into the sequence labeling framework so far due to a lack of encodings that can support reentrancy, disconnected nodes and cycles.[1]

We bridge this gap by defining sequence labeling encodings for these problems that are framed as predicting directed graphs between words - which, following some previous literature (Agić et al., 2015; Barnes et al., 2021) - we group under the umbrella term of *dependency graph parsing*. By taking dependency tree encodings as a starting point and extending them to support graphs, we define a wide range of both unbounded and bounded encodings to cast dependency graph parsing problems as sequence labeling. To test the performance of the encodings, we experiment on two dependency graph parsing tasks, representative of different kinds of linguistically-relevant structures: semantic dependency parsing (where the output are DAGs, directed acyclic graphs) and enhanced UD parsing (where graphs have cycles). The source code is available at `github.com/anaezquerro/separ`.

2 Background

Sequence-labeling approaches that partially perform syntactic parsing have long been known, in the form of supertagging (Joshi and Srinivas, 1994). Still, the goal of supertagging is to cut the search space of the parsing process, not to fully replace it:

[1]Note that there exist linearizations to implement semantic dependency parsing with sequence-to-sequence (seq2seq) approaches (Lin et al., 2022). However, seq2seq models are not to be confused with sequence labeling models. In seq2seq models, output length is arbitrary. For sequence labeling, the output needs to have exactly the same length as the input, i.e., graphs need to be encoded as one label per word. Existing seq2seq linearizations do not meet this condition, so they cannot be used for sequence labeling.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 11804–11828
November 12-16, 2024 ©2024 Association for Computational Linguistics

the generated labels (supertags) are not enough to encode a whole parse tree, and a parsing algorithm is still needed to fill the missing information.

The first attempt at addressing the full task of natural language parsing as sequence labeling was by Spoustová and Spousta (2010), who introduced a linearization for dependency parsing where the label of each word encoded the PoS tag of its head and its relative position among words with the same PoS tag. However, machine learning algorithms of the time struggled to predict such labels, leaving the practical results far behind the state of the art.

It was with the development of deep learning and its use in NLP architectures that parsing as sequence labeling became practically viable. This was shown by Gómez-Rodríguez and Vilares (2018) for constituent parsing and by Strzyz et al. (2019) for dependency parsing. For the purposes of this paper, we will leave work on constituent parsing linearizations (Kitaev and Klein, 2020; Amini and Cotterell, 2022) aside and outline the encodings that cast dependency parsing as sequence labeling, since they have a more direct relation to our target problem of dependency graph parsing and will be the inspiration of our proposed encodings.

Common notation Let V be a vocabulary of possible tokens. We will denote a sentence of length n by $w = [w_1, w_2, \ldots, w_n] \in V^n$. Let S_n be the set of possible parses (dependency trees or dependency graphs) for sentences of length n. Then, a *sequence labeling encoding* for parsing is an injective function $\mathcal{E} : S_n \to L^n$; where L is a set of labels that is defined depending on the encoding. Thus, a sequence labeling encoding is used to represent a parse for a sentence of length n, $w = [w_1, w_2, \ldots, w_n]$, as a sequence of n labels (one per word), $l = [l_1, l_2, \ldots, l_n] \in L^n$.

Since $\mathcal{E}$ is injective, one can recover a parse in S_n from its associated label sequence in L^n via the inverse function, $\mathcal{E}^{-1} : \{\mathcal{E}(s) \mid s \in S_n\} \to S_n$. This enables parsing as sequence labeling: if we train a tagger to predict a function $f_\Theta : V^n \to L^n$ (where V is the vocabulary of possible input tokens, and Θ are the model parameters) that associates each word w with the encoding of its parse, we can obtain the parse for a sentence w as $\mathcal{E}^{-1}(f_\Theta(w))$.

A theoretical limitation is that no known encodings are bijective, so $\mathcal{E}^{-1}$ is not defined on all possible sequences of labels (L^n), but just on those that correspond to encodings of valid trees. We call the set of such sequences, $\Gamma_n = \{\mathcal{E}(s) \mid s \in S_n\} \subseteq L^n$, the set of *well-formed* label sequences for length n. Since f_Θ is an approximation by a sequence labeling classifier, it is possible that it outputs ill-formed label sequences. However, this is workable in practice, since there are simple heuristics to fix ill-formed sequences converting them to well-formed ones (i.e., mapping from L^n to Γ_n).

We now define concepts related to k-planarity (Yli-Jyrä et al., 2003) to later define the coverage of various encodings. Two arcs in a dependency tree or graph are said to *cross* if their arrows cross when drawn above the words. Thus, two arcs (w_i, w_j) and (w_k, w_l) such that $\min(i,j) < \min(k,l)$ cross iff $\min(k,l) < \max(i,j) < \max(k,l)$. A tree or graph is *noncrossing*, or *1-planar*, if it contains no crossing arcs. We also introduce the term *relaxed 1-planar* for a tree or graph with no pair of crossing arcs pointing in the same direction (i.e., only opposite crossing arcs are allowed). Finally, a tree or graph is *(relaxed) k-planar* for some $k \geq 0$ if it can be written as the union of k (relaxed) 1-planar subgraphs (called planes).

Dependency parsing encodings A *sequence labeling encoding for dependency parsing* is a sequence labeling encoding where the set of parses of interest, S_n, is T_n, the set of dependency trees for sentences of length n.

In all dependency parsing encodings defined so far in the literature, each label l_i assigned to a word w_i is of the form (d_i, x_i), where d_i represents the label of the dependency going to w_i, and it is x_i that varies between encodings and encodes the unlabeled dependency tree. Thus, we will focus on x_i and ignore dependency labels from now on.

Positional encodings We call *positional encodings* those where x_i encodes the position of the head of w_i. Let w_h be the head of w_i. The simplest such encoding is the *naive positional encoding* where $x_i = h$, i.e., it encodes directly the position of the head of w_i (as in the CoNLL format). However, this encoding has been shown to not work well in practice (Strzyz et al., 2019). Instead, the *relative positional encoding* (Li et al., 2018; Strzyz et al., 2019) represents a relative offset, $x_i = h - i$. While it did not obtain good results under simpler implementations, it has been shown to be viable when coupled with more powerful language models (Vacareanu et al., 2020).

To reduce sparsity, one can use PoS tags to locate head words. In the *relative PoS-based encod-*

ing (Spoustová and Spousta, 2010; Strzyz et al., 2019), x_i is a pair (p_i, o_i) such that if $o_i > 0$ then w_h is the o_ith among the words with PoS tag p_i that are located to the right of w_i, and if $o_i < 0$, then it is the $-o_i$th among words with PoS tag p_i to the left of h_i. This encoding has been shown to be very effective in high-resource setups where high-accuracy PoS tags are avaible (Strzyz et al., 2019), but tends to suffer when this is not the case (Muñoz-Ortiz et al., 2021). It is also possible to restrict offsets using properties other than PoS tags, as in the *relative head-based* encoding of Lacroix (2019), based on tagging words as leaf or non-leaf nodes and then encoding $x_i = o_i$ and finding the o_ith non-leaf to the right or $-o_i$th to the left.

Unbounded bracketing encodings Based on the axiomatization by Yli-Jyrä and Gómez-Rodríguez (2017), these representations encode each dependency arc by adding one symbol to the label of each of its endpoints. In the simplest version, the *basic bracketing encoding* adapted to sequence labeling by Strzyz et al. (2019), a right arc from w_i to w_j is represented by including a / symbol at the label x_i and a > symbol at x_j, whereas a left arc from x_j to x_i is encoded by a < symbol at x_i and a \ symbol at x_j.[2] The label for a word is a string formed by concatenating all symbols involving that word, so that for example, a label $x_i =$ \>// means that the word w_i has one outgoing arc to the left, two to the right, and one incoming arc from the left.

Decoding is a linear-time process where the sentence is read from left to right. Two separate stacks are used: one to decode / and > into right arcs, and the other to decode < and \ into left arcs. Symbols are treated as brackets, with left (opening) brackets / and < being pushed into the stack when read, and popped when a matching right bracket (respectively, > or \) is found, while the corresponding arc is created. Since > is always matched to the closest / and < to the closest \, the encoding cannot handle trees that have crossing arcs within the same direction, being restricted to relaxed 1-planar trees.

To improve coverage, one can apply the notion of multiplanarity (Yli-Jyrä et al., 2003), dividing the dependency tree into two separate subgraphs (planes) and encoding each separately. This yields the *2-planar bracketing encoding* (Strzyz et al., 2020), an encoding with two sets of brackets: the original /, >, < and \ are used to encode arcs in the first plane as above, and additional /*, >*, <*, and * are added to encode the arcs of the second plane. The decoding of the second plane is made with two additional separate stacks (keeping the linear-time complexity), so that arcs in different planes can always cross. This makes the encoding support relaxed 2-planar trees, and thus yields over 99% coverage on a variety of tested treebanks.

We classify these encodings as unbounded because the number of possible labels is not bounded by a constant, but scales with respect to sentence length n (consider, for example, that the first word on a sentence could have any number of / between 1 and $n-1$). Positional encodings are also unbounded, although their number of possible labels is $O(n)$ while in bracketing encodings it is $O(n^2)$. In spite of this theoretical drawback, unbounded bracketing encodings empirically tend to have fewer labels than positional encodings, and they have been shown to be a solid choice in many practical scenarios (Muñoz-Ortiz et al., 2021).

Bounded bracketing encodings Gómez-Rodríguez et al. (2023) define two encodings, derived from the basic and 2-planar bracketing encodings, but where the labels are vectors of a fixed number of bits. Thus, they are bounded, as the number of possible labels is a constant.

In the *4-bit encoding*, each label x_i is of the form $b_i^0 b_i^1 b_i^2 b_i^3$, where each b_i^j is a bit: b_i^0 is true (false) if w_i is a right (left) dependent, b_i^1 is true iff w_i is the outermost right or left dependent of its parent node; and b_i^2 and b_i^3 are true iff w_i has one or more left or right dependents, respectively. While this encoding is very compact, having a total of 16 labels, it shares the drawback of the basic brackets of not supporting same-direction crossing arcs.

The *7-bit encoding* extends it using multiplanarity to support relaxed 2-planar trees by using 7 bits to represent two planes of arcs. Labels are of the form $x_i = b_i^0 \cdots b_i^6$, three more bits than the previous encoding: a bit is added to specify whether w_i is a dependent in the first or second plane, and the two bits indicating left or right dependents are split into two bits to represent the presence of such dependents in the first plane and two for the second plane. The rest of the bits retain their meaning. This encoding consistently outperformed unbounded bracketings in the experiments of (Gómez-Rodríguez et al., 2023).

[2] In (Strzyz et al., 2019, 2020), unbounded bracketing encodings are defined differently, with arcs involving w_i and w_j being encoded at labels x_{i+1} and x_j. We choose the straightforward x_i and x_j, as the reason for the first option (reducing sparsity in projective trees) is not relevant for this work.

The decoding of bounded bracketing encodings back to a tree is described in detail in (Gómez-Rodríguez et al., 2023). It works similarly to that of unbounded brackets and is also linear time, but one stack element can generate several outgoing arcs. For example, for right arcs in the first plane, we traverse the sentence from left to right. When we find a word w_i with $b_i^3 = 1$ (w_i has right dependents) we push / to the stack. If we then find a word w_j with $b_j^0 = 1$ (w_j is a right dependent), we create the arc $w_i \rightarrow w_j$, but only pop the / symbol from the stack if $b_j^1 = 1$ (w_j is the outermost dependent). The decoding for left arcs in the first plane is analogous, but in a separate pass from right to left. For the 7-bit encoding, we add two extra passes for left and right arcs in the second plane.

Transition-based encodings Gómez-Rodríguez et al. (2020) show that many transition-based parsers can yield sequence labeling encodings. Though in theory applicable to dependency graph parsing, previous results on syntactic parsing show that the systems' accuracy degrades for non-projective trees, so we will discard this approach.

Hexatagging (Amini et al., 2023) is the overall best-performing encoding known so far for dependency parsing. It is bounded and the most compact, as it represents projective trees with only 8 possible labels per word. Yet, its design makes it unlikely that an extension to graph parsing is possible, as it is based on projectivity and treeness (requiring converting dependency trees to a constituent-like representation). Thus, we will not use it here.

3 Unbounded graph encodings

Let $w = [w_1, w_2, \dots, w_n] \in V^n$ be a sentence. A dependency graph for w is a labeled, directed graph $G = (V_w, E)$ where $V_w = \{w_1, \dots, w_n\}$. Contrary to dependency trees, dependency graphs in general allow reentrancy (two or more incoming arcs to the same node) and cycles. Let G_n be the set of dependency graphs for sentences of length n. A *sequence labeling encoding for dependency graph parsing* is one where $S_n = G_n$. We next present our unbounded encodings. As in dependency parsing encodings, we assume that each label l_i is of the form (d_i, x_i), but in this case d_i is a tuple composed of the labels of the dependencies going to w_i, sorted by head position. As before, we focus on the encoding-dependent component x_i, and ignore dependency labels from now on.

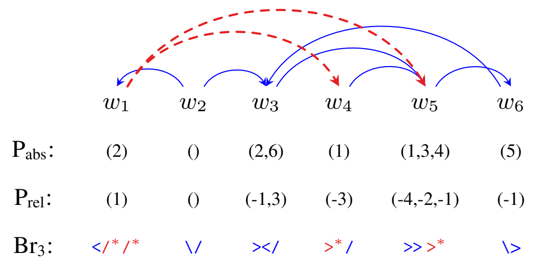


Figure 1: An example of a relaxed 2-planar dependency graph linearized with our unbounded encodings.

Positional graph encodings A naive approach for dependency graph parsing as sequence labeling is to adapt the positional encodings for dependency tree parsing. This can be done by defining x_i as a tuple of arbitrary length containing the absolute (or relative) positions of all incoming arcs of each word w_i, so x_i is an ascendingly sorted tuple with the elements of $\{h : (w_h, w_i) \in E\}$ for the naive encoding and $\{h - i : (w_h, w_i) \in E\}$ for the relative encoding. For example, in the graph in Figure 1, w_3 has incoming arcs from w_2 and w_6, so the naive encoding assigns it the tuple (2,6) and the relative encoding (-1,3). Note that our definition of a dependency graph G_n allows nodes with no incoming arcs, thus x_i might be an empty tuple.

Unbounded bracketing encodings The extension of unbounded bracketing encodings for syntactic dependency parsing to dependency graphs is straightforward. In this family of linearizations, the restriction to a single parent per node for syntactic parsing is achieved by explicitly enforcing exactly one incoming symbol, < or >, in each x_i. For graph parsing, we remove this restriction and allow more than one such symbol, as well as zero (which can even produce an empty string for disconnected nodes). The decoding process does not change, using two different stacks for right and left arcs, and keeping the linear complexity. Multiplanarity is supported by introducing new sets of brackets. In Figure 1, w_3 has one incoming arc from each direction (><) and an outgoing arc to the right (/). w_5 has three incoming arcs from the left, of which one is in the second plane (>*).

Having k sets of brackets provides coverage over relaxed k-planar graphs, like the tree encoding did for relaxed 2-planar trees, and could do for relaxed k-planar trees if more sets of brackets were added. However, it is worth noting that previous work using this encoding for syntactic parsing has never experimentally explored beyond $k = 2$ (i.e., adding one extra set of brackets <*, >*, /*, *). The rationale was the trend that most syntactic trees are 2-planar (Gómez-Rodríguez and Nivre, 2013), so complicating parsing algorithms (or encodings) did not seem worthwhile for a tiny increase in coverage. However, since dependency graphs can be denser than syntactic trees, and have less propensity to be 2-planar, we also experiment with adding a third plane with a third set of brackets <**, >**, /**, **. This gives mixed results for this particular encoding, although as will be seen later, setting $k > 2$ will prove very useful to increase accuracy in the case of bounded encodings, especially with datasets containing denser graphs.

As the plane assignment algorithm (i.e. to split arcs in gold graphs into planes in a canonical way) we extend the greedy plane assignment algorithm of (Strzyz et al., 2020) to support more than two planes: we traverse arcs in order and assign each to the lowest possible plane such that it does not cross any arcs already assigned to the same plane.

4 Bounded graph encodings

We now define two bounded encodings for graph parsing, based on the 4- and 7-bit encodings by Gómez-Rodríguez et al. (2023).

4.1 $4k$-bit encoding

Assumption This encoding assumes that the set of edges E of G can be split into k relaxed 1-planar subgraphs, such that in each subgraph, all nodes have at most one incoming arc (maximum in-degree 1). To do so, it explicitly arranges a dummy node w_0 that has dependencies towards any parentless node. Thus, all nodes in the graph $(w_1 \dots w_n)$ can be seen as having exactly one incoming arc (from a regular or the dummy node).

Encoding It uses a sequence of 4 bits to encode the arcs related to the word w_i that are in the jth subgraph. Each label x_i is a grouped sequence of $4k$ bits where the jth group of four bits encodes only the arcs of the jth subgraph. The meaning of each of the four bits in a group is as in the 4-bit encoding of Gómez-Rodríguez et al. (2023): b_i^{4j-4} is true (false) if w_i has a left (right) parent in the jth subgraph (which could be the dummy node), b_i^{4j-3} is true if w_i is the farthest dependent of its parent in the jth subgraph, b_i^{4j-2} is true if w_i has left dependents and b_i^{4j-1} is true if w_i has right dependents in the jth subgraph. Thus, this encoding concatenates k instances of said 4-bit encoding, which is injective and has coverage over relaxed 1-planar graphs with no more than one parent per node.[3]

Plane assignment We need a way to express a dependency graph as the union of k relaxed 1-planar subgraphs with at most one parent per node. The plane assignment algorithm used in the unbounded bracketing does not suffice for two reasons. First (1), arcs may need to be assigned to different subgraphs not only because they cross, but also because they have the same dependent.[4] Secondly (2), nodes that have a parent in the dependency graph may be parentless in one or more subgraphs. While this may not seem problematic because the encoding supports such nodes by linking them as children of the dummy node, this would require adding arcs that can break relaxed 1-planarity. To solve (1), we modify the plane assignment algorithm to consider two arcs incompatible if they cross or share the dependent. To solve (2), we add artificial arcs (which we call null arcs) linking each parentless node to the immediately previous node (such an arc is guaranteed to not produce a crossing). When implementing the parser, null arcs are especially labeled and excluded from the final parse. Figure 2 shows this assignment process: note the null arcs drawn with dotted lines, and the arc (w_4, w_5) being assigned to the third (green) subgraph despite not crossing any other arc, since there are already arcs going to w_5 in the other two subgraphs.

Coverage The $4k$-bit encoding has coverage over the set of dependency graphs that can be expressed as the union of k relaxed 1-planar graphs with maximum in-degree 1. This set is trivially a subset of (1) relaxed k-planar graphs, and (2) dependency graphs with maximum in-degree k. In practice, we show that $k = 2$ suffices for almost total coverage on enhanced UD datasets (Table 2), whereas on semantic dependency parsing datasets we need $k = 4$ for really high coverage in most cases (Table 1).

Decoding It is done with the same linear-time algorithm as for bounded dependency parsing encodings (Section 2), with one pass per group of bits (i.e. subgraph) and arc direction, i.e., $2k$ passes.

[3] The original 4-bit encoding is described as having coverage over relaxed 1-planar forests. This is because the task, about tree parsing, forbids cycles. The encoding itself does support graphs with cycles as long as there is no reentrancy.

[4] $4k$-bit allows only one incoming arc per node and subgraph, whose direction is encoded in the first bit of each group.

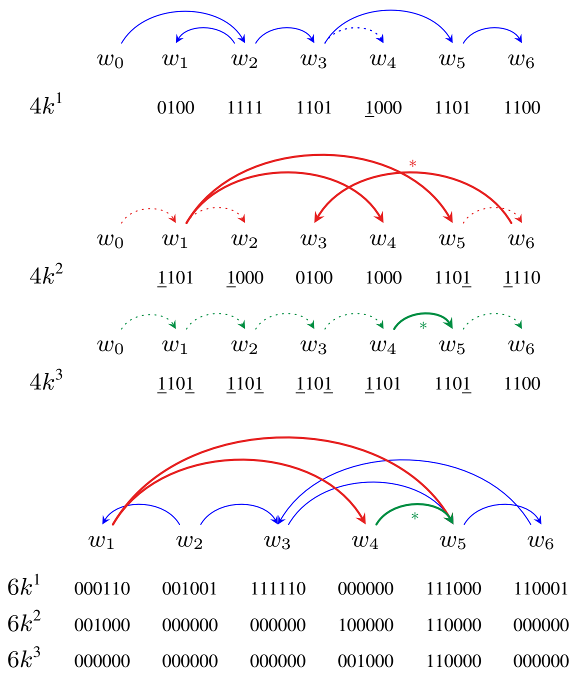


Figure 2: Bounded encodings for the example of Figure 1. The *relaxed* 1-planar subgraphs for the $4k$-bit-encoding are shown with their linearization, added null arcs are drawn with dotted lines, and their associated bits underlined. For $6k$-bit, we use colors to distinguish the subgraph pairs. Note that, in both cases, those arcs that are assigned to different planes w.r.t. the unbounded bracketing encoding (Figure 1) are marked with $*$.

4.2 $6k$-bit encoding

Assumption This encoding assumes that the parse's edge set E can be split into $2k$ relaxed 1-planar subgraphs, under two conditions. First, k subgraphs have all of their arcs pointing to the left, while in the other k, all arcs are rightward. We consider that the $2k$ subgraphs are arranged in pairs, such that each pair has a leftward subgraph and a rightward subgraph. Second, all subgraphs have maximum in-degree 1. $6k$-bit does not require dummy arcs, contrary to the $4k$-bit encoding.

Encoding Each label x_i has k groups of 6 bits each. The jth group in each label encodes information about the jth pair of subgraphs (composed of the jth rightward subgraph and the corresponding jth leftward subgraph). In particular, the meaning of each of the bits in the jth group is as follows: b_i^{6j-6} is true if w_i has a parent in the jth rightward subgraph, b_i^{6j-5} is true if w_i is the farthest dependent of its parent in said subgraph, and b_i^{6j-4} is true if w_i has at least one dependent in said subgraph. Finally, b_i^{6j-3}, b_i^{6j-2} and b_i^{6j-1} have the same meaning for the leftward subgraph.

Plane assignment To perform the assignment of arcs to subgraphs, left arcs and right arcs are processed separately (with the goal of choosing the pair whose leftward or rightward subgraph we need to assign them to). For each of these subsets, we run a modification of the plane assignment algorithm of the unbounded bracketing. Two arcs are incompatible if they cross or share the dependent (for the same reason as in the $4k$-bit encoding: subgraphs need to have maximum in-degree 1). However, here we do not need to add any null arcs, because this encoding supports representing parentless nodes natively: it suffices to set b_i^{6j-6} or b_i^{6j-3} to indicate that w_i is parentless in the jth rightward (resp. leftward) subgraph. Figure 2 contains an example of this assignment, with the same graph as in previous examples. Colors depict subgraph pairs (with individual subgraphs being the subsets of leftward and rightward arcs of each color). The assignment of pairs is different from the assignment of subgraphs for $4k$-bit: apart from not needing null arcs, the arc from w_3 to w_6 can now be assigned to the first subgraph pair (vs. the second subgraph in $4k$-bit) since arcs in different directions that share a dependent can coexist in the different subgraphs of a same pair.

Coverage $6k$-bit covers the set of graphs that can be split into $2k$ subgraphs meeting the above conditions. This is a subset of (1) relaxed k-planar graphs (each subgraph is relaxed 1-planar, so each subgraph pair is also relaxed 1-planar since joining a leftward graph with a rightward graph cannot generate crossings of arcs in the same direction, and the graph is the union of k such pairs); and (2) graphs with maximum in-degree $2k$, since each of the $2k$ subgraphs can contribute one parent to a given node. The encoding cannot cover graphs where a node has more than k incoming arcs from the same direction – even if the in-degree does not surpass $2k$ – as we only have k rightward (leftward) subgraphs. Still, as can be seen in Tables 1 and 2, the coverage is larger than that of the more compact $4k$-bit encoding for the same value of k.

Decoding Again done with the same linear-time algorithm as for bounded dependency tree encodings (Section 2), with $2k$ passes, one per subgraph.

5 Model architecture

Let w be our input sentence. The model produces a sequence of vectors $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]$ using a generic encoder E_θ. This encoder can range from lookup tables mapping words to static embeddings[5] to encoders that deeply contextualize words. We then use a generic decoder, D_ϕ, to make output predictions at the word level. The decoder could also vary widely, from simple feed-forward networks to more sophisticated architectures. Let $\mathbf{X} = [\mathbf{x}_1, \ldots \mathbf{x}_n] = D_\phi(\mathbf{W})$ be the output representations for each word. We use these outputs to predict each component x_i from the label $l_i = (d_i, x_i)$. The sequence $(x_1, ..., x_n)$ is fed to the corresponding encoding-specific decoding algorithm to recover the set of predicted arcs $\hat{E}$. Then, to predict the label representing the relationship between the pair of nodes of a predicted arc $(w_h, w_i) \in \hat{E}$, we concatenate their representations $[(\mathbf{w}_h|\mathbf{w}_i) : (w_h, w_i) \in \hat{E}]$ and use them to feed another generic decoder D_φ that predicts the dependency type (an element of d_i) associated with them.

6 Experiments

For evaluation, we use two kinds of dependency graph structures with different properties: semantic dependency parsing, and enhanced UD parsing.

Datasets For semantic dependency parsing, we will use the following datasets and formalisms from the SemEval 2018 Task 18 (Oepen et al., 2015): (i) the English dataset annotated with DELPH-IN MRS-Derived Bi-Lexical Dependencies (DM Ivanova et al., 2012) (ii), the English and Chinese datasets with Enju Predicate–Argument Structures (PAS Miyao et al., 2005), and (iii) the English and Czech datasets with Prague Semantic Dependencies (PSD Hajič et al., 2012). They are all collections of sentences annotated as graphs, where some tokens do not contribute to the graph, some might have just one parent, and others multiple parents. These representations are DAGs, excluding the study of relevant phenomena involving cycles. To study cycles, we rely on enhanced universal dependencies, particularly on the version released together with the IWPT 2021 Shared Task (Bouma et al., 2021). We evaluated five languages: Arabic, Finnish, French, Slovak, and Tamil, joining the different treebanks of the same language to build a larger annotated corpus. In Appendix A.2, we provide additional information and statistics for both DAG and IWPT treebanks.

[5]However, previous work showed that continuous, contextualized representations are needed for accurate outputs.

Metrics We use the SDP evaluation toolkit[6] (Oepen et al., 2015). We report both unlabeled and labeled F1 score (LF, UF) w.r.t. the predicted dependencies, i.e. triplets of the form (predicate, role, argument).[7] Results with further metrics, including exact match, are in the Appendix.

Models' setup For the encoders of our taggers, we use a few representatives. We focus on experiments using two Transformer-based encoders: (1) XLM-RoBERTa (Conneau et al., 2020), as a single multilingual pre-trained encoder for non-English experiments, and (2) XLNet (Yang et al., 2019) for English experiments. In addition, we experimented with biLSTMs for comparison in terms of speed-accuracy trade-off. The decoder is a 1-layered feed-forward network followed by a softmax. We did a minor hyperparameter search to tune the optimizer and batch size to our resources.[8] For comparison, we include an external model (Biaf), a biaffine semantic dependency parser (Dozat and Manning, 2018) - from `supar`[9] - using the same encoders as for our experiments.

6.1 Empirical results

We provide here the main results of our experiments. Supplementary data and insights can be found in Appendix A.4.

For brevity, we refer to absolute and relative graph positional encodings as A and R. Unbounded bracketing encodings are denoted as B_k, where k is the number of supported planes. For $4k$-bit and $6k$-bit encodings, we use $\mathrm{B4}_k$ and $\mathrm{B6}_k$, respectively, where the subindex is the value of k (number of subgraphs or pairs the graph is divided into).

Table 1 shows the results for the DAG experiments, including DM, PAS, and PSD in-domain test data, for all our encodings. The results on the out-of-domain test and development sets are in the Appendix (Table 7), but the trends are similar.

Positional encodings yield, overall, less accurate results for dependency graph parsing, possibly influenced by the added complexity of representing a list of head positions, rather than just one as in syntactic parsing, into a single label. The rest of our encodings, though, are much more robust: most of them are rougly on par or even outperform the

[6]github.com/semantic-dependency-parsing/toolkit

[7]Detection of root nodes is considered as identifying additional virtual dependencies, and counts for evaluation.

[8]See Appendix A.3 for more details about the model size and training specifications.

[9]github.com/yzhangcs/parser

	DM$_{en}$			**PAS$_{en}$**			**PSD$_{en}$**			**PSD$_{cs}$**			**PAS$_{zh}$**		
	UF	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**
A	88.66	87.94	100	86.66	85.29	100	89.56	79.19	100	90.04	85.33	100	77.12	74.63	100
R	91.92	91.23	100	90.29	88.86	100	89.74	79.39	100	89.60	84.92	100	79.26	76.96	100
B$_2$	95.16	94.46	99.94	**95.82**	**94.31**	99.98	92.31	81.80	99.83	92.75	88.14	99.83	87.66	85.37	99.98
B$_3$	94.63	93.75	100	95.73	94.21	100	92.33	81.65	99.99	92.74	88.02	100	**87.73**	**85.42**	100
B4$_2$	86.45	85.84	91.23	79.84	78.82	83.18	**92.87**	81.96	99.68	92.88	88.24	99.65	77.53	75.47	86.01
B4$_3$	92.64	91.64	97.96	89.65	88.23	93.40	92.68	81.99	99.96	93.11	88.33	99.98	83.60	81.34	93.42
B4$_4$	95.07	94.35	99.64	93.79	92.35	97.59	92.80	82.00	100	**93.39**	**88.79**	100	81.40	78.68	96.60
B6$_2$	91.21	90.67	96.37	87.70	86.64	91.59	92.66	81.88	99.77	93.37	88.54	99.79	81.88	79.81	92.09
B6$_3$	94.90	94.15	99.51	93.58	92.16	97.56	92.61	**82.13**	99.98	93.44	88.61	99.98	85.67	83.50	96.57
B6$_4$	**95.23**	**94.52**	99.96	95.32	93.87	99.37	92.74	81.89	100	93.30	88.45	100	87.06	84.77	98.30
Biaf	95.07	94.31	100	95.69	94.12	100	92.95	82.08	100	93.65	88.73	100	87.80	85.49	100

Table 1: DAG in-distribution results. OF is the coverage of each encoding in terms of oracle F-score.

	ar			**fi**			**fr**			**sk**			**ta**		
	UF	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**
A	75.00	69.17	100	84.52	80.97	100	80.79	76.58	100	83.53	79.39	100	34.22	27.27	100
R	82.06	75.58	100	85.90	82.34	100	83.06	78.66	100	86.79	82.59	100	64.69	53.69	100
B$_2$	87.85	80.98	99.82	91.19	88.16	99.60	91.06	87.60	99.97	93.31	90.15	99.79	73.62	62.02	100
B$_3$	87.82	81.22	99.94	91.20	88.13	99.94	90.59	86.65	100	93.81	90.44	99.96	73.62	62.02	100
B4$_2$	87.84	81.21	99.77	91.53	88.46	99.60	92.37	88.64	99.87	93.79	90.41	99.72	75.81	63.36	99.95
B4$_3$	87.81	81.11	99.90	91.58	88.41	99.87	92.60	88.88	99.98	94.08	90.51	99.94	76.01	**64.54**	100
B4$_4$	88.09	81.27	99.94	91.64	88.56	99.94	92.61	88.47	100	93.85	90.32	99.99	76.01	64.54	100
B6$_2$	87.68	80.97	99.85	92.02	89.03	99.69	92.60	88.82	99.93	94.06	90.86	99.85	**76.16**	63.48	100
B6$_3$	88.10	81.37	99.93	91.64	88.71	99.89	91.66	88.06	99.99	**94.26**	**90.94**	99.97	76.16	63.48	100
B6$_4$	**88.33**	**81.60**	99.95	**92.03**	**89.07**	99.95	**92.96**	**89.80**	100	93.89	90.73	99.99	76.16	63.48	100
Biaf	89.70	82.48	100	93.54	90.93	100	93.71	89.77	100	94.22	90.61	100	76.08	65.74	100

Table 2: EUD parsing results on IWPT datasets. Notation as in Table 1.

biaffine parser, a competitive baseline. The relative performance of different bracketings strongly depends on whether bounded encodings obtain high coverage. In the English and Czech PSD datasets, which are comparatively less dense, bounded encodings achieve almost total theoretical coverage (column OF on the table) and excel in performance, and the more compact $4k$-bit encodings seem to be slightly better than $6k$-bit encodings. However, in PAS datasets, where the coverage of bounded encodings is lower due to higher graph density (see Appendix Table 4 for graph density statistics), unbounded brackets clearly outperform bounded brackets, and $4k$-bit encodings suffer more than $6k$-bit ones, due to having even less coverage. The DM dataset is a middle ground, both in terms of coverage and results, with unbounded, B4$_4$ and B6$_4$ obtaining fairly similar accuracies.

With respect to the parameter k (i.e., number of subgraphs or pairs used in the bracketing encodings), it is important to increase it beyond 2 for bounded encodings in the denser datasets, where coverage and accuracy increase hand in hand when setting k to 3 and 4. This differs w.r.t. common practice in encodings for dependency tree parsing, where values of k beyond 2 are not used in experiments due to $k = 2$ providing almost full coverage. The case for $k > 2$ is less clear for unbounded bracketing, where $k = 2$ already provides fairly good coverage. B_3 is still better than B_2 in one dataset, but this might be due to statistical noise.

For EUD parsing (Table 2), in spite of the presence of cycles, graphs are sparser than in our DAG datasets, so all encodings have fairly good coverage. In this context, bounded encodings obtain the best performance, consistent with the DAG results, although in this case they fall somewhat short of the biaffine parser for some languages, being on par in others. $6k$-bit encodings generally outperform $4k$-bit encodings. The $4k$-bit encoding performs better (at least in terms of labelled F-score) for Tamil, maybe due to the much smaller training set favoring compact encodings. The absolute and relative encodings (prone to sparsity problems) obtain even poorer results on Tamil.

Finally, with respect to the parameter k, the effect of increasing it beyond 2 in the EUD datasets is not as clear as in denser datasets, which makes sense as the increases in coverage afforded by larger values of k is minimal. However, there seems to be a clearly positive (if weak) influence in the case of bounded encodings (where the best results are always achieved by encodings with k at least 3, except in the case of Tamil, where the value of k mostly does not matter as all but one encoding reach 100% coverage). This suggests that increasing k might be useful even when the coverage increase is small. On the other hand, for unbounded bracketings, no clear trend is visible, with B_3 outperforming B_2 in some datasets and underperforming it in others.

6.2 Speed analysis

In Figure 3, we present a Pareto front in terms of UF and speed (tokens per second) across three datasets: DM (English, in-distribution) and two IWPT corpora (Arabic and French). In Appendix A.4 we include additional figures for other treebanks to provide a more complete picture.

Alongside the biaffine baseline, we include a lighter encoder, a two-stacked biLSTM. Note that, for space reasons, we did not include biLSTM results in the previous section, as the UF performance is substantially lower, as seen in the figure. The 2-layered BiLSTM is faster than the pretrained encoders in all cases, followed by XLM-R and XLNet. Although BiLSTMs are less accurate, they can still appear on the Pareto front in some setups.

Overall, our parsers offer either similar or faster speeds compared to the biaffine model, with the exception of those relying on the $4k$-bit encoding, which consistently lag behind the rest of the models across all datasets. This may be due to the extra dependencies generated in the encoding step (Section 4.1), which force the decoder to process more arcs in the decoding step and predict more dependency types. The positional encodings (A, R), instead, offer the best inference speed.[10]

7 Conclusion

For the first time, we have framed dependency graph parsing tasks, like semantic dependency or EUD parsing, as sequence labeling tasks. We have proposed a wide variety of bounded and unbounded encodings that – with the right representation – can be learned by standard encoders. Among unbounded encodings, positional strategies performed poorly, but bracketing-based encoders obtained robust performance, excelling especially in dense datasets. On the other hand, the more compact bounded encodings, with a fixed number of bits per label, obtained the best results in sparser datasets. Overall, results are comparable or even outperform a strong biaffine baseline. Thus, dependency graph parsing can effectively be solved as sequence labeling, as both bounded and unbounded encodings are learnable using standard bidirectional encoders and simple feed-forward decoders.

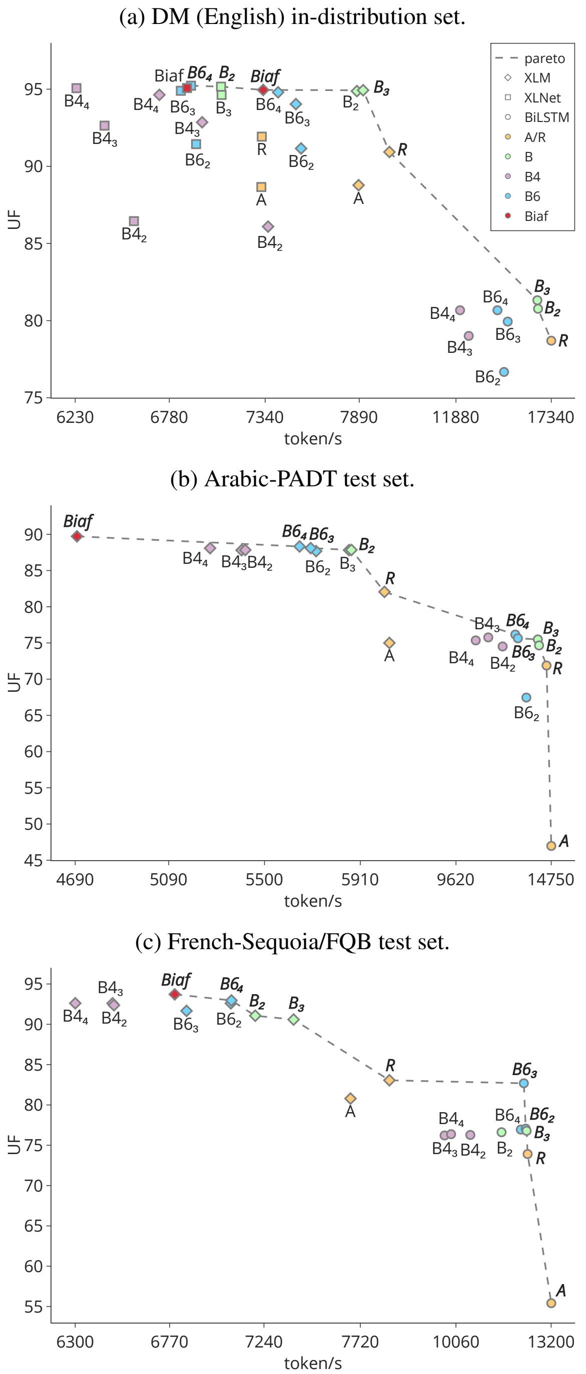


Figure 3: Pareto front: UF vs. speed. X-axis rescaled, outliers omitted for clarity.

[10]Our decoding process is parallel for positional encodings, since each label x_i can be independently processed to create a subset of predicted arcs, thus optimizing the inference time.

Limitations

Anchoring This work focuses on dependency graph parsing, i.e. structured prediction problems where the input is a sentence and the output is a graph where nodes correspond to words. While this template fits a considerable range of tasks, including several flavors of semantic dependency parsing, EUD parsing or graph-based sentiment analysis; there are kinds of meaning representation parsing that do not fit this framework. More in detail, meaning representations can be hierarchically organized in different formal flavors for semantic graphs, as described by Oepen et al. (2019, 2020). These *flavors* refer to the relationship between the words of a sentence and the nodes of the graph (known as *anchoring*). The scope of our work includes flavor (0) representations. In this flavor, the nodes of the graphs are the tokens of the input sentence, meaning there is a one-to-one correspondence between the nodes of the graph and the words. In future work, we aim to generalize our encodings to more relaxed flavors. However, we would like to remark that the technical contributions to casting these flavors as sequence labeling would mainly arise from aspects other than the linearizations of the graphs. Once the nodes of such graphs are computed, our linearizations could be directly applied to any dependency-based formalism.

Physical resources Our computational resources are limited to eight shared RTX 3090 GPUs. Despite this, we have successfully trained models for various formalisms and languages. This has given us insight into the learnability of different encodings. While more powerful architectures might boost our results, our extensive empirical findings clearly support the key contributions of our work.

Ethical considerations

We do not observe ethical implications in our work. Our research focuses on improving the technical aspects of semantic dependency parsing and enhanced Universal Dependencies parsing, which are primarily computational and linguistic challenges. The methodologies and applications discussed do not involve sensitive personal data, human subjects, or scenarios that could lead to ethical concerns. Thus, our findings and techniques can be applied within the field without ethical reservations.

We acknowledge the environmental impact associated with the training process in neural models, particularly in terms of CO_2 emissions, with Spain being the country where the experiments were conducted. We measured the carbon footprint per epoch of our models at training and inference time: 0.24 g CO_2 (training) and 0.06 g CO_2 (inference) for the larger pretrained architectures, and 0.06 g CO_2 (training) and 0.02 g CO_2 (inference) for the non-pretrained architectures. These metrics are relatively low compared with the latest models in NLP research, encouraging future work to consider energy-efficient approaches. Additionally, for reference, the European Union sets a maximum of around 115 g CO_2 per kilometer for newly manufactured cars.

Acknowledgments

We acknowledge the European Research Council (ERC), which has funded this research under the Horizon Europe research and innovation programme (SALSA, grant agreement No 101100615). We also acknowledge grants SCANNER-UDC (PID2020-113230RB-C21) funded by MICIU/AEI/10.13039/501100011033; GAP (PID2022-139308OA-I00) funded by MICIU/AEI/10.13039/501100011033/ and ERDF, EU; LATCHING (PID2023-147129OB-C21) funded by MICIU/AEI/10.13039/501100011033 and ERDF, EU; and TSI-100925-2023-1 funded by Ministry for Digital Transformation and Civil Service and "NextGenerationEU" PRTR; as well as funding by Xunta de Galicia (ED431C 2024/02), and Centro de Investigación de Galicia "CITIC", funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centres of the Galician University System (CIGUS).

References

Željko Agić, Alexander Koller, and Stephan Oepen. 2015. Semantic dependency graph parsing using tree approximations. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 217–227, London, UK. Association for Computational Linguistics.

Afra Amini and Ryan Cotterell. 2022. On parsing as tagging. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8884–8900, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Afra Amini, Tianyu Liu, and Ryan Cotterell. 2023. Hexatagging: Projective dependency parsing as tagging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1453–1464, Toronto, Canada. Association for Computational Linguistics.

Jeremy Barnes, Robin Kurtz, Stephan Oepen, Lilja Øvrelid, and Erik Velldal. 2021. Structured sentiment analysis as dependency graph parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3387–3402, Online. Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2021. From raw text to enhanced Universal Dependencies: The parsing shared task at IWPT 2021. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 146–157, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and Joakim Nivre. 2013. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4):799–845.

Carlos Gómez-Rodríguez, Diego Roca, and David Vilares. 2023. 4 and 7-bit labeling for projective and non-projective dependency trees. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6375–6384, Singapore. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Michalina Strzyz, and David Vilares. 2020. A unifying theory of transition-based and sequence labeling parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3776–3793, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom? a contrastive study of syntacto-semantic dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea. Association for Computational Linguistics.

Aravind K. Joshi and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Nikita Kitaev and Dan Klein. 2020. Tetra-tagging: Word-synchronous parsing with linear-time inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6255–6261, Online. Association for Computational Linguistics.

Ophélie Lacroix. 2019. Dependency parsing as sequence labeling with head-based encoding and multitask learning. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 136–143, Paris, France. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Boda Lin, Zijun Yao, Jiaxin Shi, Shulin Cao, Binghao Tang, Si Li, Yong Luo, Juanzi Li, and Lei Hou. 2022. Dependency parsing via sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7339–7353, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2005. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn treebank. In *Natural Language Processing–IJCNLP 2004: First International Joint Conference,* *Hainan Island, China, March 22-24, 2004, Revised Selected Papers 1*, pages 684–693. Springer.

Alberto Muñoz-Ortiz, Michalina Strzyz, and David Vilares. 2021. Not all linearizations are equally data-hungry in sequence labeling parsing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 978–988, Held Online. INCOMA Ltd.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*, pages 1–22, Online. Association for Computational Linguistics.

Stephan Oepen, Omri Abend, Jan Hajic, Daniel Hershcovich, Marco Kuhlmann, Tim O'Gorman, Nianwen Xue, Jayeol Chun, Milan Straka, and Zdenka Uresova. 2019. MRP 2019: Cross-framework meaning representation parsing. In *Proceedings of the Shared Task on Cross-Framework Meaning Representation Parsing at the 2019 Conference on Natural Language Learning*, pages 1–27, Hong Kong. Association for Computational Linguistics.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Drahomíra Spoustová and Miroslav Spousta. 2010. Dependency parsing as a sequence labeling task. *The Prague Bulletin of Mathematical Linguistics*, 94:7.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. 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)*, pages 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2020. Bracketing encodings for 2-planar dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2472–2484, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Robert Vacareanu, George Caique Gouveia Barbosa, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2020. Parsing as tagging. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5225–5231, Marseille, France. European Language Resources Association.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *Preprint*, arXiv:1505.00853.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Anssi Yli-Jyrä and Carlos Gómez-Rodríguez. 2017. Generic axiomatization of families of noncrossing graphs in dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1745–1755, Vancouver, Canada. Association for Computational Linguistics.

Anssi Yli-Jyrä et al. 2003. Multiplanarity–a model for dependency structures in treebanks. In *TLT 2003. Proceedings of the second workshop on treebanks and linguistic theories*, volume 9, pages 189–200.

A Appendix

A.1 Behavior in degenerate cases

All of the encodings presented reduce to existing dependency tree encodings in degenerate cases. In the case of the positional and unbounded bracketing encoding, when using them to encode a corpus containing only trees, the result will be as with the positional and unbounded tree encodings (the former will always generate tuples of size one, which are like the labels of the corresponding tree encoding, and the latter will generate the exact same labels with at most one incoming arc symbol as the unbounded bracketing encoding for trees).

In the case of $4k$-bit bracketing, if $k = 1$, the encoding reduces exactly to the existing 4-bit encoding for projective trees. Even if one used $k > 1$ for a corpus of projective trees, the result would be isomorphic, as extra planes would not be used so every group of bits beyond the first would always take the same values.

Finally, the $6k$-bit bracketing for $k = 1$ also becomes isomorphic to the 4-bit encoding: even though there are six bits, when the graph is a tree there are only four possible combinations of b_i^0, b_i^1, b_i^3 and b_i^4 (which vary depending on whether a node is a left or right dependent and whether it is the outermost dependent, with combinations that make it have two or zero parents excluded) and these are isomorphic to the four possible combinations of b_i^0 and b_i^1 in the 4-bit encoding, so there is a structure-preserving bijection between the labels generated by both encodings. The only caveat is that one

	train	dev	test
ar	PADT	PADT	PADT
fi	TDT	TDT	TDT, PUD
fr	Sequoia	Sequoia	Sequoia, FQB
sk	SNK	SNK	SNK
ta	TTB	TTB	TTB

Table 3: Multilingual benchmark obtained from joining the different treebanks from the IWPT 2021 datasets (Bouma et al., 2021).

needs to consider the root node as outermost right dependent of a dummy root node w_0.

This means that, except for the change in root node handling for the $6k$-bit encoding, there is no extra complexity added, and no possible change in accuracy,[11] if one uses the encodings presented here to train a model on a treebank of projective trees.

A.2 Treebank statistics

The treebanks released in the IWPT 2021 Shared Task and used in this work (Bouma et al., 2021) were joined (preserving the original train, validation and test split) to train each language in a single data benchmark, as specified in Table 3. In Table 4, we summarize some general statistics about the datasets used to train and evaluate our models. Table 5 shows a summary of the labels generated by our encodings for each dataset used.

A.3 Experimental setup

Our linearization systems are conformed by two modules: a neural encoder $E_\theta : \mathbb{R}^{n\times d_x} \to \mathbb{R}^{n\times d_h}$ and a neural decoder with two different submodules: $D_\phi : \mathbb{R}^{d_h} \to L$ and $D_\varphi : \mathbb{R}^{2\cdot d_h} \to \mathcal{R}$.[12] Given a sentence $(w_1, ..., w_n)$, the encoder E_θ outputs their contextualized representations $(\mathbf{w}_1, ..., \mathbf{w}_n)$. Then, D_ϕ produces a label component $x_i \in \mathcal{X}$ for each hidden representation $\mathbf{w}_i$, and the decoding process recovers the set of predicted arcs $\hat{E} \subseteq \{(w_h, w_i) : h, i \in [1, n], h \neq i\}$. Finally, to obtain the type of dependency relation associated to each arc, D_φ concatenates the word contextualizations of each predicted arc $(\mathbf{w}_h, \mathbf{w}_i)$ and outputs the dependency type $r_{h,i} \in \mathcal{R}$. Note that in this formulation, the component d_i of the tuple $l_i = (d_i, x_i)$ introduced in Section 2 is not directly predicted with D_ψ, but the dependency relation associated to each decoded arc. One could define the component d_i by sorting the elements of the set $\{r_{h,i} : (w_h, w_i) \in \hat{E}\}$ by h, and use D_ψ to predict the entire sorted sequence. In practice, our implementation does not reconstruct d_i as we just use each obtained $r_{h,i}$ to directly label the corresponding dependency. Figure 4 shows an illustration of the inference steps to reconstruct the labeled semantic graph for a given sentence.

[11]This is guaranteed in practice because unused labels cannot harm accuracy in any way, as sequence labeling systems only train on the labels that they effectively see in the training set, not on the theoretical space of labels.

[12]Here we denote with $\mathcal{R}$ the set of dependency relationships between the edges of the graph.

	#sents	% planes 1	2	3	4	5	h/n	d/n	a/g	len	#cycs.
DAG											
DM$_{en}$	33964	57.44	41.39	1.16	0.01	0.00	0.79	0.75	17.68	22.52	0
	1692	59.69	39.30	1.00	0.00	0.00	0.79	0.75	17.55	22.28	0
	1410	53.19	44.96	1.77	0.07	0.00	0.78	0.74	17.60	22.66	0
	1849	59.17	40.02	0.81	0.00	0.00	0.77	0.73	13.11	17.08	0
PAS$_{en}$	33964	53.46	45.88	0.67	0.00	0.00	1.02	0.98	22.96	22.52	0
	1692	58.81	40.43	0.77	0.00	0.00	1.01	0.97	22.53	22.28	0
	1410	51.21	48.09	0.71	0.00	0.00	1.02	0.98	23.15	22.66	0
	1849	56.14	43.16	0.70	0.00	0.00	1.02	0.96	17.35	17.08	0
PSD$_{en}$	33964	58.27	36.02	5.03	0.61	0.06	0.70	0.67	15.80	22.52	0
	1692	59.10	35.64	4.37	0.65	0.18	0.71	0.68	15.74	22.28	0
	1410	55.89	38.01	5.39	0.71	0.00	0.70	0.67	15.79	22.66	0
	1849	64.85	28.39	5.90	0.76	0.11	0.68	0.64	11.57	17.08	0
PSD$_{cs}$	40047	55.67	37.72	5.86	0.69	0.06	0.78	0.75	18.34	23.45	0
	2010	58.56	35.62	5.42	0.30	0.10	0.78	0.75	17.97	22.99	0
	1670	55.81	38.32	5.39	0.48	0.00	0.77	0.74	17.71	22.99	0
	5226	66.84	26.48	5.59	0.92	0.15	0.78	0.74	13.19	16.82	0
PAS$_{zh}$	25896	65.56	33.51	0.92	0.00	0.00	1.02	0.98	22.95	22.43	0
	2440	61.52	36.89	1.60	0.00	0.00	1.02	0.99	28.60	27.95	0
	8976	64.43	34.44	1.14	0.00	0.00	1.02	0.98	24.47	23.89	0
IWPT											
ar	6075	63.80	31.23	4.13	0.67	0.15	1.06	1.04	39.22	36.85	1386
	909	68.10	27.06	4.29	0.33	0.22	1.06	1.03	35.24	33.27	225
	680	62.65	32.35	3.82	1.03	0.15	1.06	1.04	44.11	41.56	178
fi	12217	64.89	27.58	6.61	0.78	0.09	1.08	1.01	14.41	13.33	1855
	1364	62.83	28.89	7.55	0.37	0.29	1.09	1.01	14.57	13.42	203
	2555	68.96	25.28	5.17	0.51	0.04	1.06	0.99	15.24	14.44	414
fr	2231	69.03	29.05	1.79	0.13	0.00	1.05	1.01	23.77	22.64	546
	412	67.72	30.83	1.21	0.24	0.00	1.05	1.01	25.51	24.28	112
	2745	90.02	9.40	0.58	0.00	0.00	1.02	0.95	12.76	12.45	193
sk	8483	73.00	25.49	1.44	0.05	0.02	1.05	0.95	9.96	9.50	469
	1060	70.85	26.98	2.08	0.09	0.00	1.06	0.98	12.70	12.01	105
	1061	67.01	29.88	2.83	0.19	0.09	1.07	0.98	12.79	12.00	117
ta	400	89.50	10.50	0.00	0.00	0.00	1.02	0.96	16.19	15.82	1
	80	87.50	12.50	0.00	0.00	0.00	1.06	0.99	16.68	15.79	22
	120	91.67	8.33	0.00	0.00	0.00	1.04	0.98	17.29	16.57	38

Table 4: Number of sentences (**#sents**) and cycles (**#cycs.**); ratio of $\{1...5\}$-planar graphs (**%planes**); average number of heads (**h/n**) and dependants per node (**d/n**), arcs per graph (**a/g**) and sentence length (**len**). Each subrow corresponds to the train, dev, i.d. and o.o.d (test) split of each treebank in the DAG (IWPT) dataset.

As a pretrained encoder, we primary finetuned XLM-RoBERTa-R (Conneau et al., 2020), and XLNet (Yang et al., 2019), for experiments in English. Additionally, we conducted experiments with a non-pretrained 2-layer BiLSTM encoder. We excluded these results from the main content of the paper, except for reporting accuracy-speed trade-offs, due to space constraints. However, we included them in Appendix A.4 for a more complete picture. The pretrained encoders were only fed with word in-

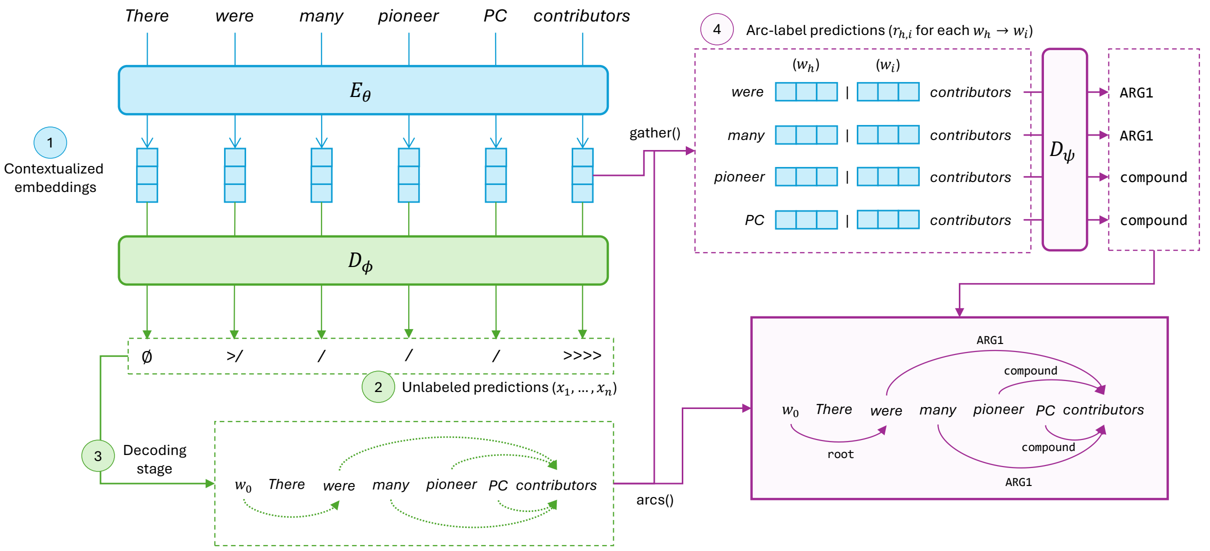


Figure 4: Prediction steps of our neural parser for the sentence "*There were many pioneer PC contributors*" (from the $\mathrm{DM_{en}}$ treebank) using the 1-planar bracketing encoding.

	A	R	B_2	B_3	$B4_2$	$B4_3$	$B4_4$	$B6_2$	$B6_3$	$B6_4$	$\lvert\mathcal{R}\rvert$
DAG											
$\mathrm{DM_{en}}$	12666	7601	622	714	159	526	882	252	486	621	60
	2226	1249	257	273	116	285	426	144	231	280	44
	1959	1133	269	292	116	279	406	148	226	263	44
	1606	926	288	307	120	293	416	156	246	281	48
$\mathrm{PAS_{en}}$	23664	17434	1510	1590	159	587	1254	212	543	907	43
	3041	1936	449	464	117	320	550	111	239	364	40
	2792	1842	476	488	116	320	561	128	263	401	42
	2421	1633	539	555	122	372	643	121	279	441	42
$\mathrm{PSD_{en}}$	2424	2978	1302	1890	210	677	986	360	677	819	91
	617	702	436	533	144	299	356	185	265	283	81
	570	647	401	500	146	286	340	166	237	256	78
	574	715	406	501	146	299	366	168	239	261	75
$\mathrm{PSD_{cs}}$	3768	4676	2155	3057	225	864	1323	413	861	1050	62
	743	885	550	671	159	347	427	216	305	323	62
	712	840	570	693	156	338	401	196	281	300	58
	1107	1465	822	1084	191	537	705	275	460	523	65
$\mathrm{PAS_{zh}}$	26749	21334	1603	1747	158	567	1128	189	474	774	33
	6040	4237	660	699	129	374	644	144	307	438	31
	13557	10259	1105	1197	143	467	896	163	377	606	32
IWPT											
ar	2543	2368	872	1218	149	349	468	217	368	429	1056
	714	657	363	425	106	180	209	122	160	174	356
	917	742	362	417	106	174	206	119	149	166	363
fi	1630	1593	1192	1729	185	537	745	226	415	505	417
	528	484	468	590	143	292	342	147	216	239	191
	485	526	517	677	150	318	381	152	215	238	234
fr	877	780	593	652	119	200	254	130	168	188	47
	369	331	297	305	91	137	158	85	110	119	45
	406	336	349	369	101	161	185	105	130	141	44
sk	422	510	501	605	154	318	363	149	217	232	266
	244	271	277	306	121	200	213	109	137	139	187
	287	325	287	322	125	206	236	118	143	150	163
ta	129	133	112	112	61	65	65	33	33	33	117
	94	94	79	79	54	61	61	29	29	29	63
	119	117	78	78	48	57	58	26	26	26	77

Table 5: Number of labels ($|\mathcal{X}|$) and dependency types ($|\mathcal{R}|$) produced with our encodings in each treebank.

formation, excluding PoS-tags or character embeddings from the input, and a feed-forward network was stacked in the last layer of the architecture to reduce the original output dimension. For the decoder, we relied on a 1-layered feed-forward network that parameterizes each part (D_ϕ and D_φ) of the module.

Hyp.	Model configuration
d_h	400
d_w	400
d_p	100
d_c	30
Hyp.	**Training configuration**
epochs	200 (BiLSTM), 200 (pretrained)
batch size	2000 (BiLSTM), 200 (pretrained)
η	$1 \cdot 10^{-3}$ (BiLSTM), $5 \cdot 10^{-5}$ (XLM-R), $1 \cdot 10^{-5}$ (XLNet)

Table 6: Model and training hyperparameters.

The full system was trained end-to-end with the AdamW optimizer (Loshchilov and Hutter, 2019), setting the learning rate η to $1 \cdot 10^{-3}$ or $1 \cdot 10^{-4}$ for non-pretrained models and $5 \cdot 10^{-5}$ or $1 \cdot 10^{-5}$ for the pretrained architectures. All layers use a LeakyReLU (Xu et al., 2015) with a negative slope of 0.01 as the activation function and the latent space applies a dropout of 0.33. Table 6 summarizes the hyperparameter selection to configure our neural models.

A.4 Results

We report here more detailed results. Table 7 presents the results for the out-of-domain DAG datasets, with trends similar to those observed for in-domain data in Table 1. Tables 8 to 17 show more detailed results for each dataset. Metrics include both labeled and unlabeled F1 score (LF, UM), w.r.t. the predicted dependency (predicate, role, argument triplets); as well as labeled and unlabeled exact match, known as LM and UM. Additionally, we include the performance in terms of (i) tagging accuracy and ratio of well-formed trees in Table 18, (ii) number of planes in Table 19 and (iii) number of cycles in Table 20. Finally, Figure 5 summarizes the rest of accuracy-speed comparisons for the treebanks not included in Section 6.2.

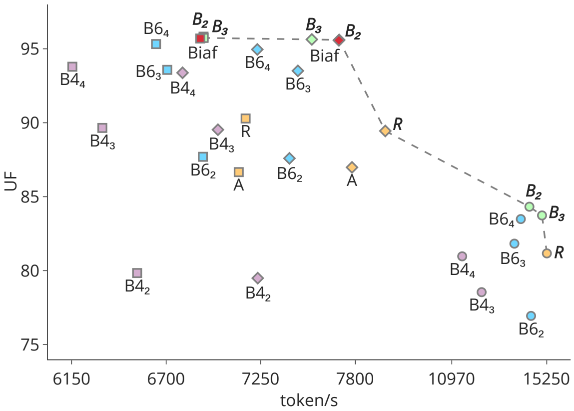
	$\mathbf{DM_{en}}$			$\mathbf{PAS_{en}}$			$\mathbf{PSD_{en}}$			$\mathbf{PSD_{cs}}$		
	UF	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**	**UF**	**LF**	**OF**
A	87.47	86.35	100	86.22	84.62	100	88.55	78.24	100	86.92	71.78	100
R	89.94	88.90	100	90.14	88.51	100	88.47	78.26	100	86.43	71.43	100
B_2	92.56	91.54	99.97	**94.82**	**93.18**	99.96	91.45	81.38	99.82	89.69	74.89	99.79
B_3	91.92	90.77	100	94.77	93.12	100	91.60	81.48	99.98	89.91	75.15	99.96
$B4_2$	85.54	84.71	93.58	81.37	80.18	86.16	91.44	80.97	99.65	90.11	75.22	99.59
$B4_3$	89.97	88.56	98.57	89.71	88.02	94.78	91.90	81.41	99.95	89.97	75.25	99.94
$B4_4$	92.41	91.28	99.75	93.03	91.40	98.09	91.73	81.34	99.99	90.13	75.19	99.97
$B6_2$	89.09	88.24	97.27	88.31	87.01	93.35	91.78	81.35	99.77	**90.44**	**75.72**	99.74
$B6_3$	92.05	91.05	99.66	93.27	91.70	98.24	91.88	81.53	99.96	90.40	75.61	99.95
$B6_4$	**92.71**	**91.67**	99.97	94.34	92.64	99.53	**91.95**	**81.61**	99.99	90.34	75.56	99.97
Biaf	92.52	91.43	100	94.13	92.39	100	91.81	81.27	100	90.63	75.44	100

Table 7: DAG out-of-distribution results. Notation as in Table 1.

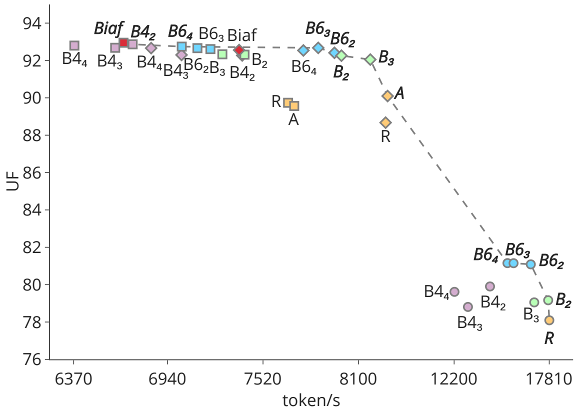
	dev				**in-distribution**				**out-of-distribution**			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	59.38	4.61	57.07	4.33	57.85	6.98	54.45	6.38	60.15	5.32	57.99	4.73
R	78.70	8.87	75.73	7.23	74.18	12.49	69.79	10.49	79.56	10.22	76.74	8.81
B_2	80.77	**13.76**	78.36	**12.34**	**75.83**	**15.14**	71.93	**12.82**	81.81	**15.25**	79.25	**13.65**
B_3	**81.32**	13.26	**78.84**	11.49	75.57	13.90	72.01	11.68	**82.06**	13.53	**79.58**	11.88
$B4_2$	73.11	4.47	71.03	4.04	69.71	8.98	67.07	7.79	73.27	4.43	71.15	4.08
$B4_3$	79.01	8.16	76.56	6.95	73.99	12.28	70.70	11.20	79.49	8.27	77.10	7.33
$B4_4$	80.67	10.78	78.21	9.86	75.28	13.90	72.12	12.17	81.54	13.65	78.96	12.12
$B6_2$	76.67	7.45	74.51	6.60	72.31	12.28	69.26	11.03	77.22	8.16	74.97	7.39
$B6_3$	79.94	10.78	77.54	9.57	74.97	14.76	71.70	12.33	80.57	12.94	78.22	10.99
$B6_4$	80.67	11.63	78.11	10.07	75.77	14.71	**72.35**	12.60	81.32	13.24	78.78	11.52
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	88.78	41.70	87.91	38.94	87.01	46.46	85.70	43.32	88.94	41.78	88.14	39.24
R	90.93	37.66	90.03	35.11	88.67	42.56	87.43	40.13	91.41	39.42	90.62	37.29
B_2	94.88	**57.23**	94.05	**52.48**	91.74	52.73	90.57	49.86	95.25	59.93	94.46	**56.26**
B_3	**94.93**	56.17	**94.12**	51.35	**92.08**	**54.35**	**90.83**	50.68	95.23	**60.05**	94.46	55.91
$B4_2$	86.09	9.86	85.47	9.50	85.33	23.20	84.48	22.34	85.85	10.11	85.18	9.63
$B4_3$	92.86	33.83	92.08	32.13	90.57	42.73	89.46	40.83	93.01	33.98	92.22	32.15
$B4_4$	94.63	51.63	93.78	47.87	91.58	51.43	90.42	48.24	94.98	54.26	94.16	50.83
$B6_2$	91.16	22.27	90.50	20.92	88.73	35.48	87.83	33.86	91.02	22.46	90.39	21.51
$B6_3$	94.03	46.67	93.22	43.05	91.46	49.32	90.24	46.78	94.31	49.65	93.51	45.98
$B6_4$	94.80	55.96	94.03	51.06	92.02	53.98	90.75	**50.84**	**95.31**	59.28	**94.51**	55.20
Biaf	94.95	55.18	94.15	51.56	92.21	52.03	91.00	48.51	95.54	59.22	94.86	55.50
XLNet	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	88.66	41.63	87.94	39.43	87.47	47.92	86.35	45.27	88.67	40.78	88.01	38.95
R	91.92	39.79	91.23	37.09	89.94	46.19	88.90	44.19	92.15	41.67	91.44	39.78
B_2	95.16	58.01	94.46	54.54	92.56	53.98	91.54	51.11	95.29	60.52	94.58	56.62
B_3	94.63	54.61	93.75	49.43	91.92	53.06	90.77	49.43	94.84	57.80	94.05	54.08
$B4_2$	86.45	10.43	85.84	9.93	85.54	23.36	84.71	22.28	86.14	10.70	85.52	10.22
$B4_3$	92.64	31.77	91.64	29.01	89.97	39.48	88.56	36.07	92.61	31.62	91.69	29.37
$B4_4$	95.07	53.62	94.35	50.35	92.41	53.27	91.28	49.76	95.21	55.14	94.48	51.77
$B6_2$	91.44	22.70	90.87	22.06	89.07	35.48	88.22	34.07	91.46	24.11	90.90	23.23
$B6_3$	94.90	51.49	94.15	47.94	92.05	53.33	91.05	50.57	94.87	52.13	94.20	48.82
$B6_4$	**95.23**	**59.15**	**94.52**	**55.53**	**92.71**	**55.33**	**91.67**	**52.51**	**95.5**	**61.7**	**94.83**	**58.27**
Biaf	95.07	55.39	94.31	52.06	92.52	52.46	91.43	50.03	95.20	54.85	94.52	51.89
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B_2	99.95	98.16	99.95	98.16	99.97	99.19	99.97	99.19	99.97	99.00	99.97	99.00
B_3	100	99.93	100	99.93	100	100	100	100	100	100	100	100
$B4_2$	90.45	13.83	90.45	13.83	92.22	33.59	92.22	33.59	89.77	12.00	89.77	12.00
$B4_3$	97.64	52.55	97.64	52.55	98.07	70.69	98.07	70.69	97.32	49.00	97.32	49.00
$B4_4$	99.56	87.94	99.56	87.94	99.61	92.21	99.61	92.21	99.45	85.05	99.45	85.05
$B6_2$	96.09	32.91	96.09	32.91	96.65	54.46	96.65	54.46	95.68	31.15	95.68	31.15
$B6_3$	99.44	83.62	99.44	83.62	99.52	89.89	99.52	89.89	99.27	79.85	99.27	79.85
$B6_4$	99.95	98.51	99.95	98.51	99.95	98.97	99.95	98.97	99.93	97.93	99.93	97.93

Table 8: Performance in the DM dev, in-distribution and out-of-distribution sets. OC is the oracle coverage and UF (LF) and UM (LM) are the unlabeled (labeled) F-score and exact match. The best performing model is highlighted in bold.

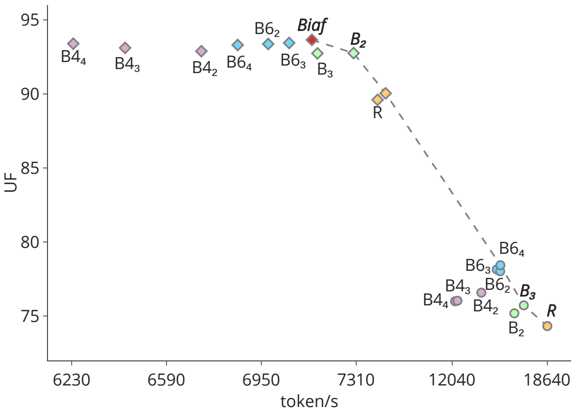
(a) PAS (English) in-distribution set.



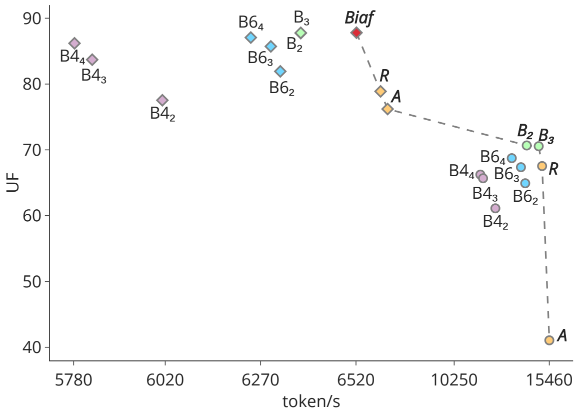
(b) PSD (English) in-distribution set.



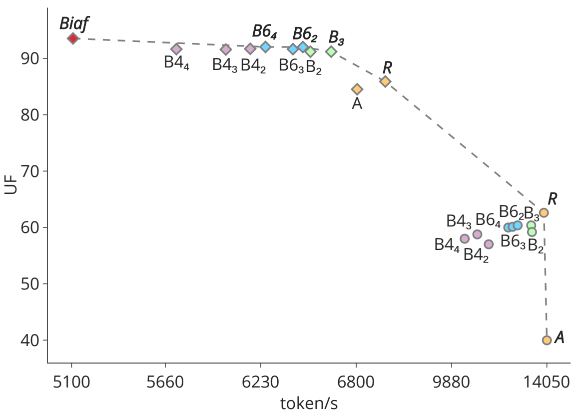
(c) PSD (Czech) test set.



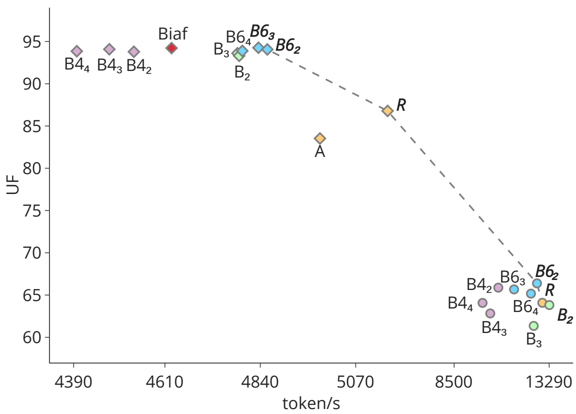
(d) PAS (Chinese) test set.



(e) Finnish-TDT/PUD test set.



(f) Slovak-SNK test set.



(g) Tamil-TTB test set.

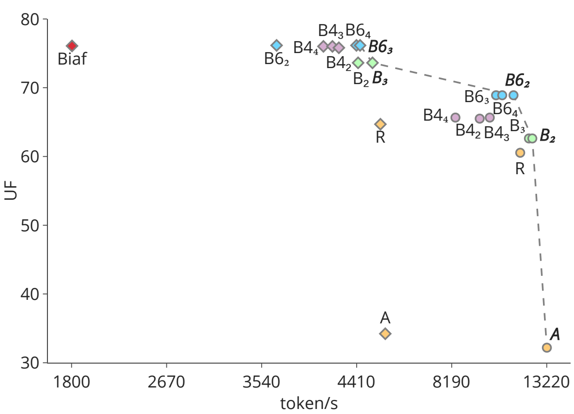


Figure 5: Pareto front for the rest of treebanks.

	dev				in-distribution				out-of-distribution			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	65.52	4.40	63.88	4.33	63.07	6.65	60.58	6.33	65.89	5.61	64.00	5.32
R	81.17	6.81	79.04	6.03	77.51	11.79	74.68	11.25	81.86	9.22	79.83	8.87
B_2	**84.32**	**13.33**	**82.39**	**12.7**	**80.32**	**17.36**	**77.59**	**16.39**	**84.93**	**16.43**	**83.02**	**15.37**
B_3	83.73	12.41	81.86	11.56	79.73	16.23	76.87	15.36	84.43	14.89	82.50	14.24
$B4_2$	69.78	1.21	68.08	1.13	68.85	4.87	66.50	4.76	70.18	2.31	68.54	2.31
$B4_3$	78.54	5.46	76.77	5.25	75.32	10.49	72.81	10.11	79.08	5.56	77.30	5.38
$B4_4$	80.97	7.59	78.89	7.02	77.18	12.49	74.66	12.22	81.36	8.69	79.45	8.57
$B6_2$	76.94	3.69	75.13	3.48	75.18	9.57	72.63	9.46	77.56	4.61	75.80	4.49
$B6_3$	81.83	7.66	80.09	7.45	78.90	15.04	76.41	14.12	82.35	10.28	80.48	9.69
$B6_4$	83.49	11.84	81.57	11.35	79.51	17.09	76.64	15.95	84.02	13.83	82.08	13.30
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	86.98	29.43	85.49	26.60	86.15	39.70	84.32	36.78	87.78	32.39	86.20	27.96
R	89.45	27.45	87.95	23.90	88.47	36.40	86.71	33.80	89.83	29.73	88.08	25.30
B_2	95.60	55.18	94.05	47.16	94.29	55.87	92.50	50.73	95.74	55.97	**94.15**	**46.63**
B_3	**95.63**	**55.96**	**94.14**	**48.3**	**94.38**	**56.35**	**92.67**	**51.16**	**95.79**	**56.38**	94.08	46.63
$B4_2$	79.49	2.34	78.37	2.13	81.22	8.82	79.87	8.60	79.72	4.14	78.52	3.61
$B4_3$	89.53	11.63	88.21	10.99	89.63	23.96	87.93	22.93	89.56	14.54	88.05	13.00
$B4_4$	93.38	30.07	91.79	26.31	92.47	41.48	90.69	38.18	93.48	30.38	91.89	26.36
$B6_2$	87.59	7.38	86.45	6.95	88.13	18.33	86.70	17.79	87.67	9.04	86.51	8.04
$B6_3$	93.51	28.58	92.14	25.89	92.99	42.08	91.35	39.05	93.56	31.15	91.97	26.77
$B6_4$	94.96	46.24	93.46	39.72	93.72	52.62	91.98	48.40	95.25	47.75	93.69	40.31
Biaf	95.57	48.37	94.04	40.78	94.38	54.57	92.56	49.32	95.70	47.81	94.09	40.90
XLNet	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	86.66	29.29	85.29	26.52	86.22	40.62	84.62	37.75	86.55	31.26	84.97	27.13
R	90.29	29.57	88.86	26.03	90.14	40.56	88.51	37.70	90.98	32.09	89.45	28.07
B_2	**95.82**	**55.96**	**94.31**	**48.72**	**94.82**	58.57	**93.18**	53.11	96.02	55.85	94.44	**47.58**
B_3	95.73	55.32	94.21	47.09	94.77	**59.28**	93.12	**53.76**	**96.17**	**56.56**	**94.62**	47.52
$B4_2$	79.84	2.34	78.82	2.13	81.37	9.14	80.18	8.98	79.77	4.02	78.64	3.78
$B4_3$	89.65	11.42	88.23	10.64	89.71	24.07	88.02	22.50	89.74	15.19	88.30	13.83
$B4_4$	93.79	30.57	92.35	27.66	93.03	43.59	91.40	40.56	93.71	31.38	92.15	27.42
$B6_2$	87.70	7.45	86.64	7.09	88.31	18.28	87.01	17.74	88.04	8.98	86.91	8.27
$B6_3$	93.58	28.65	92.16	26.03	93.27	42.51	91.70	38.99	93.72	31.32	92.26	27.72
$B6_4$	95.32	47.38	93.87	41.84	94.34	55.38	92.64	50.84	95.41	48.82	93.91	42.32
Biaf	95.69	47.02	94.12	40.50	94.13	52.41	92.39	47.86	95.67	46.45	94.07	39.60
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B_2	99.98	99.29	99.98	99.29	99.98	99.30	99.98	99.30	99.98	99.23	99.98	99.23
B_3	100	100	100	100	100	100	100	100	100	100	100	100
$B4_2$	82.46	2.70	82.46	2.70	85.13	10.38	85.13	10.38	82.40	4.26	82.40	4.26
$B4_3$	92.89	14.75	92.89	14.75	94.13	32.29	94.13	32.29	92.87	18.32	92.87	18.32
$B4_4$	97.27	44.54	97.27	44.54	97.70	62.74	97.70	62.74	97.22	47.04	97.22	47.04
$B6_2$	91.28	9.93	91.28	9.93	92.76	24.12	92.76	24.12	91.36	11.52	91.36	11.52
$B6_3$	97.38	42.55	97.38	42.55	97.97	63.33	97.97	63.33	97.45	46.45	97.45	46.45
$B6_4$	99.27	78.30	99.27	78.30	99.39	87.29	99.39	87.29	99.28	80.08	99.28	80.08

Table 9: Performance in the PAS (English) dataset. Notation as in Table 8.

	dev				in-distribution				out-of-distribution			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	65.53	13.40	52.51	4.75	65.09	20.93	50.21	9.57	65.97	12.41	53.15	5.38
R	78.09	15.11	62.44	5.60	74.78	23.69	58.79	10.55	79.01	16.84	63.98	7.09
B_2	79.17	16.52	64.83	5.96	76.42	24.99	61.99	11.47	80.30	17.26	66.45	7.74
B_3	79.05	15.39	64.24	5.39	76.25	23.15	61.13	11.03	79.88	16.19	65.31	6.44
$B4_2$	79.90	17.09	66.36	6.10	76.19	25.74	62.75	12.71	80.64	17.67	68.25	7.56
$B4_3$	78.81	17.16	65.00	5.82	76.04	25.15	62.44	12.44	80.01	17.26	67.09	7.51
$B4_4$	79.62	16.38	65.57	5.11	76.43	26.07	62.80	12.17	80.39	17.55	67.35	7.39
$B6_2$	81.09	18.79	**66.78**	6.81	77.85	27.85	**63.37**	**13.25**	**81.97**	**21.28**	**68.7**	**8.87**
$B6_3$	**81.15**	**20.07**	66.11	6.45	77.96	28.12	62.67	12.93	81.46	20.15	67.11	7.21
$B6_4$	81.15	19.72	66.38	**6.88**	**78.04**	**28.45**	62.63	13.20	81.47	20.33	67.00	7.56
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	90.10	43.33	79.14	15.39	87.97	49.59	77.43	24.28	91.06	46.28	80.76	18.38
R	88.67	35.89	78.14	14.61	87.08	45.27	76.76	24.34	89.85	38.48	80.03	16.61
B_2	92.26	47.80	81.60	16.88	91.19	53.27	81.09	**28.29**	93.32	50.53	83.21	18.32
B_3	92.05	46.10	81.30	16.60	90.86	51.76	80.69	27.85	93.19	51.36	83.09	**19.44**
$B4_2$	92.27	45.60	81.41	16.24	91.39	52.46	80.90	26.66	93.20	50.71	82.93	18.14
$B4_3$	92.30	47.87	81.51	16.24	91.60	54.57	81.11	27.10	93.42	51.48	83.31	19.44
$B4_4$	92.66	48.16	81.77	17.66	91.31	52.68	80.83	25.74	93.55	52.13	83.31	19.09
$B6_2$	92.41	46.74	81.73	16.67	91.48	53.16	81.05	26.61	**93.64**	**52.54**	**83.38**	19.44
$B6_3$	**92.67**	**48.37**	**81.79**	**18.01**	91.43	53.16	81.15	26.61	93.55	51.95	83.24	18.79
$B6_4$	92.54	48.16	81.62	16.38	**91.85**	**54.73**	**81.63**	28.02	93.50	52.13	83.16	19.03
Biaf	92.56	42.62	81.73	16.31	91.28	48.84	80.50	25.15	93.40	45.57	83.46	18.50
XLNet	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	89.56	43.62	79.19	15.82	88.55	52.30	78.24	26.55	90.57	46.28	80.63	18.03
R	89.74	38.65	79.39	15.46	88.47	48.78	78.26	26.28	90.49	41.19	80.44	17.55
B_2	92.31	47.52	81.80	**18.01**	91.45	52.62	81.38	26.88	93.45	51.65	83.36	19.03
B_3	92.33	47.94	81.65	16.60	91.60	54.46	81.48	26.93	93.60	51.54	83.40	18.38
$B4_2$	**92.87**	48.51	81.96	16.95	91.44	52.89	80.97	26.66	93.76	51.83	**83.58**	19.09
$B4_3$	92.68	**49.08**	81.99	16.88	91.90	54.73	81.41	26.34	**93.83**	52.54	83.30	18.79
$B4_4$	92.80	48.65	82.00	17.09	91.73	54.52	81.34	27.20	93.60	52.66	83.40	19.09
$B6_2$	92.66	47.73	81.88	17.02	91.78	53.54	81.35	26.34	93.67	52.84	83.54	**19.92**
$B6_3$	92.61	48.30	**82.13**	17.59	91.88	**55.44**	81.53	**27.85**	93.50	52.60	83.38	18.79
$B6_4$	92.74	48.23	81.89	16.95	**91.95**	54.73	**81.61**	27.53	93.59	**53.25**	83.30	18.62
Biaf	92.95	45.46	82.08	16.74	91.81	49.92	81.27	25.04	93.86	46.75	83.77	18.20
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B_2	99.73	93.90	99.73	93.90	99.56	93.24	99.56	93.24	99.72	94.74	99.72	94.74
B_3	99.98	99.29	99.98	99.29	99.94	99.13	99.94	99.13	99.94	99.11	99.94	99.11
$B4_2$	99.58	91.63	99.58	91.63	99.42	92.75	99.42	92.75	99.53	91.49	99.53	91.49
$B4_3$	99.95	98.79	99.95	98.79	99.89	98.54	99.89	98.54	99.92	98.76	99.92	98.76
$B4_4$	100	99.93	100	99.93	99.97	99.62	99.97	99.62	99.97	99.65	99.97	99.65
$B6_2$	99.69	92.98	99.69	92.98	99.60	94.54	99.60	94.54	99.66	92.67	99.66	92.67
$B6_3$	99.96	98.94	99.96	98.94	99.90	98.81	99.90	98.81	99.93	98.94	99.93	98.94
$B6_4$	100	99.93	100	99.93	99.98	99.68	99.98	99.68	99.98	99.65	99.98	99.65

Table 10: Performance in the PSD (English) dataset. Notation as in Table 8.

	dev				in-distribution				out-of-distribution			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	56.59	8.02	48.84	4.55	53.72	12.78	38.43	7.02	58.19	8.21	50.62	4.73
R	74.32	11.86	63.72	6.11	68.16	15.98	48.51	**8.02**	76.36	11.69	66.70	6.57
B$_2$	75.18	10.30	65.30	5.45	67.94	14.33	48.78	7.16	76.95	11.79	67.73	6.37
B$_3$	75.72	10.96	66.35	5.81	68.74	14.87	50.21	7.54	77.68	11.64	68.67	6.22
B4$_2$	76.58	11.92	66.89	6.05	68.48	16.34	50.68	7.96	77.79	13.38	69.23	7.26
B4$_3$	76.03	12.99	66.50	5.81	67.95	15.63	49.73	7.67	77.37	13.23	68.75	7.06
B4$_4$	76.00	12.75	66.12	5.99	68.64	15.92	50.25	7.44	77.20	12.29	68.45	6.57
B6$_2$	78.02	**14.43**	68.10	**6.95**	70.44	17.24	**51.74**	8.00	79.58	**15.77**	70.63	**8.21**
B6$_3$	78.14	13.47	67.93	6.41	**70.62**	**17.49**	51.35	7.88	79.65	14.28	70.30	6.87
B6$_4$	**78.43**	14.07	**68.52**	6.47	70.44	16.88	51.44	7.48	**79.71**	14.68	**70.76**	7.31
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	90.04	41.38	85.33	25.45	86.92	43.93	71.78	15.38	91.13	42.44	86.74	26.47
R	89.60	35.03	84.92	24.01	86.43	40.20	71.43	15.33	90.80	37.71	86.52	24.53
B$_2$	92.75	44.55	88.14	28.38	89.69	47.51	74.89	18.16	93.92	47.56	89.83	31.24
B$_3$	92.74	46.05	88.02	29.16	89.91	48.34	75.15	18.70	93.75	46.97	89.71	31.89
B4$_2$	92.88	45.03	88.24	28.74	90.11	48.95	75.22	18.68	94.06	47.81	89.86	30.20
B4$_3$	93.11	46.71	88.33	28.56	89.97	47.91	75.25	18.27	94.25	50.00	**90.12**	**32.49**
B4$_4$	93.39	47.25	**88.79**	29.88	90.13	49.00	75.19	19.15	94.00	49.00	89.86	31.04
B6$_2$	93.37	47.90	88.54	**30.36**	**90.44**	**50.33**	**75.72**	**19.48**	94.18	48.66	90.07	31.39
B6$_3$	**93.44**	**48.2**	88.61	30.18	90.40	50.00	75.61	19.38	**94.27**	**50.65**	90.10	32.29
B6$_4$	93.30	47.07	88.45	29.28	90.34	49.67	75.56	18.87	94.13	50.05	89.89	32.34
Biaf	93.65	42.22	88.73	26.59	90.63	42.79	75.44	16.04	94.67	44.88	90.36	29.30
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	99.77	94.13	99.77	94.13	99.57	93.32	99.57	93.32	99.76	94.18	99.76	94.18
B$_3$	99.99	99.52	99.99	99.52	99.94	98.91	99.94	98.91	99.98	99.60	99.98	99.60
B4$_2$	99.53	88.44	99.53	88.44	99.30	90.49	99.30	90.49	99.55	89.85	99.55	89.85
B4$_3$	99.96	98.92	99.96	98.92	99.90	98.53	99.90	98.53	99.95	99.05	99.95	99.05
B4$_4$	99.99	99.76	99.99	99.76	99.97	99.60	99.97	99.60	99.98	99.85	99.98	99.85
B6$_2$	99.72	92.63	99.72	92.63	99.56	92.94	99.56	92.94	99.73	93.18	99.73	93.18
B6$_3$	99.97	99.22	99.97	99.22	99.93	98.91	99.93	98.91	99.97	99.35	99.97	99.35
B6$_4$	99.99	99.82	99.99	99.82	99.98	99.69	99.98	99.69	99.99	99.85	99.99	99.85

Table 11: Performance in the PSD (Czech) dataset. Notation as in Table 8.

	dev				in-distribution			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	41.06	5.72	39.56	5.66	39.31	8.40	37.80	8.20
R	67.54	7.79	65.01	7.49	67.13	9.71	64.47	9.51
B$_2$	**70.67**	11.41	**68.58**	11.27	68.87	**11.8**	66.54	**11.72**
B$_3$	70.55	**11.81**	68.41	**11.7**	**69.06**	11.76	**66.72**	11.68
B4$_2$	61.11	3.64	59.21	3.62	59.52	2.50	57.51	2.50
B4$_3$	65.68	7.64	63.64	7.59	64.29	7.17	62.09	7.05
B4$_4$	66.24	7.42	64.18	7.38	64.94	6.80	62.74	6.80
B6$_2$	64.92	7.82	63.02	7.77	63.51	7.05	61.49	7.01
B6$_3$	67.36	8.92	65.38	8.85	66.09	8.20	63.94	8.11
B6$_4$	68.74	10.20	66.77	10.05	67.26	8.57	65.09	8.44
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	76.20	21.72	74.02	20.01	75.41	18.61	73.08	17.50
R	78.89	16.22	76.66	15.60	79.13	15.66	76.65	14.96
B$_2$	87.74	36.52	85.39	33.30	87.47	31.27	84.93	28.44
B$_3$	**87.78**	**36.6**	**85.5**	**33.39**	**87.81**	**32.42**	**85.3**	**29.3**
B4$_2$	77.54	4.90	75.53	4.79	77.82	4.06	75.63	3.65
B4$_3$	83.70	14.55	81.46	13.83	83.97	13.11	81.52	12.25
B4$_4$	86.19	23.83	83.91	22.06	86.47	20.08	84.02	18.36
B6$_2$	81.92	12.14	79.87	11.73	82.22	10.61	79.94	10.08
B6$_3$	85.72	23.51	83.54	21.93	85.88	19.22	83.53	17.62
B6$_4$	87.06	30.96	84.77	28.41	87.02	25.74	84.59	23.32
Biaf	87.80	3.74	85.49	3.70	87.79	5.94	85.23	5.94
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	99.97	98.86	99.97	98.86	99.97	98.40	99.97	98.40
B$_3$	100	100	100	100	100	100	100	100
B4$_2$	87.34	6.23	87.34	6.23	88.25	5.25	88.25	5.25
B4$_3$	94.20	23.96	94.20	23.96	94.98	22.91	94.98	22.91
B4$_4$	97.12	49.82	97.12	49.82	97.68	49.22	97.68	49.22
B6$_2$	92.58	17.65	92.58	17.65	93.22	15.70	93.22	15.70
B6$_3$	96.98	46.79	96.98	46.79	97.35	43.69	97.35	43.69
B6$_4$	98.61	71.40	98.61	71.40	98.85	69.51	98.85	69.51

Table 12: Performance in the PAS (Chinese) dataset. Notation as in Table 8.

	dev				test			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	46.97	5.59	42.18	4.56	51.70	1.87	46.45	1.21
R	71.88	6.03	63.73	4.85	71.30	2.53	63.46	1.32
B$_2$	74.66	6.03	67.32	4.56	73.86	3.08	66.57	1.98
B$_3$	75.49	**7.21**	68.46	5.44	74.27	3.41	67.26	1.76
B4$_2$	74.52	6.18	66.82	4.71	73.25	3.41	65.80	**2.31**
B4$_3$	75.77	6.18	68.57	5.00	74.37	3.41	67.12	1.98
B4$_4$	75.35	6.76	67.97	5.00	74.14	3.30	67.06	1.87
B6$_2$	67.46	2.21	56.52	1.47	66.35	1.32	55.66	0.77
B6$_3$	75.64	7.21	68.26	**5.74**	74.23	3.74	67.37	2.09
B6$_4$	**76.16**	6.91	**68.97**	5.44	**74.69**	**4.4**	**67.67**	2.09
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	75.00	15.29	69.17	9.41	80.57	12.65	74.45	4.95
R	82.06	11.76	75.58	7.65	81.41	7.81	75.51	3.96
B$_2$	87.85	19.71	80.98	10.88	86.53	17.05	80.28	7.70
B$_3$	87.82	19.56	81.22	10.88	86.63	17.49	80.62	7.37
B4$_2$	87.84	21.18	81.21	**11.91**	87.32	18.26	80.99	8.25
B4$_3$	87.81	20.00	81.11	11.62	87.40	18.15	81.13	7.92
B4$_4$	88.09	20.59	81.27	11.62	87.25	18.59	80.92	7.48
B6$_2$	87.68	21.47	80.97	11.32	87.12	18.48	80.86	7.92
B6$_3$	88.10	20.59	81.37	11.32	87.15	19.14	81.02	**8.8**
B6$_4$	**88.33**	**22.35**	**81.6**	10.88	**87.52**	**19.8**	**81.21**	8.80
Biaf	89.72	17.65	82.69	7.94	88.80	17.16	82.38	7.37
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	99.82	95.00	99.82	95.00	99.85	95.16	99.85	95.16
B$_3$	99.94	98.82	99.94	98.82	99.98	99.45	99.98	99.45
B4$_2$	99.77	91.62	99.77	91.62	99.80	94.17	99.80	94.17
B4$_3$	99.90	97.94	99.90	97.94	99.94	98.68	99.94	98.68
B4$_4$	99.94	99.41	99.94	99.41	99.97	99.23	99.97	99.23
B6$_2$	99.85	95.59	99.85	95.59	99.88	96.81	99.88	96.81
B6$_3$	99.93	98.68	99.93	98.68	99.96	98.79	99.96	98.79
B6$_4$	99.95	99.56	99.95	99.56	99.98	99.45	99.98	99.45

Table 13: Performance in the IWPT (Arabic) dataset. Notation as in Table 8.

	dev				test			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	39.97	5.83	33.29	3.33	41.89	10.41	34.84	7.48
R	**62.6**	**9.43**	**49.33**	3.56	**62.93**	**14.81**	49.87	**8.06**
B$_2$	59.19	7.36	48.40	**4.11**	58.82	11.14	47.96	7.77
B$_3$	60.37	8.10	48.60	3.95	61.06	12.02	49.41	7.70
B4$_2$	57.01	6.38	46.09	3.13	57.68	9.46	46.44	6.01
B4$_3$	58.76	7.44	47.79	3.84	58.77	10.56	47.71	6.82
B4$_4$	58.00	5.79	48.54	3.60	58.36	9.24	48.79	7.26
B6$_2$	60.37	6.97	47.54	3.33	61.60	10.92	48.34	6.82
B6$_3$	60.11	7.51	49.11	3.91	61.06	11.14	**49.88**	7.04
B6$_4$	59.99	7.40	47.86	3.52	60.64	10.85	48.46	6.30
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	84.52	39.33	80.97	27.04	83.25	45.75	79.57	32.11
R	85.90	33.58	82.34	23.99	86.18	40.54	82.76	29.84
B$_2$	91.19	47.01	88.16	35.69	91.01	54.47	88.13	43.18
B$_3$	91.20	46.77	88.13	34.99	91.27	55.43	88.28	44.21
B4$_2$	91.70	46.97	88.60	35.07	91.43	54.25	88.42	42.08
B4$_3$	91.58	46.58	88.41	34.52	91.66	56.31	88.49	43.55
B4$_4$	91.64	47.44	88.56	35.58	91.43	54.55	88.48	43.18
B6$_2$	92.02	49.00	89.03	37.30	91.74	56.23	88.90	44.50
B6$_3$	91.64	**49.08**	88.71	**37.53**	92.04	58.06	**89.28**	45.82
B6$_4$	**92.03**	48.18	**89.07**	36.75	**92.15**	**58.36**	89.27	**46.19**
Biaf	93.54	49.67	90.93	37.53	93.78	57.70	91.36	46.77
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	99.60	94.25	99.60	94.25	99.41	91.72	99.41	91.72
B$_3$	99.94	99.41	99.94	99.41	99.91	99.27	99.91	99.27
B4$_2$	99.60	94.17	99.60	94.17	99.40	92.67	99.40	92.67
B4$_3$	99.87	98.83	99.87	98.83	99.80	98.09	99.80	98.09
B4$_4$	99.94	99.41	99.94	99.41	99.92	98.97	99.92	98.97
B6$_2$	99.69	96.01	99.69	96.01	99.57	94.94	99.57	94.94
B6$_3$	99.89	99.02	99.89	99.02	99.84	98.53	99.84	98.53
B6$_4$	99.95	99.49	99.95	99.49	99.94	99.34	99.94	99.34

Table 14: Performance in the IWPT (Finnish) dataset.

	dev				test			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	55.41	11.91	48.92	5.14	43.33	8.74	40.80	7.04
R	73.91	8.82	64.81	6.01	75.00	10.92	69.98	8.50
B$_2$	76.62	19.09	67.18	4.95	78.92	11.41	74.04	9.95
B$_3$	76.79	17.70	67.96	5.79	78.94	11.16	74.05	10.19
B4$_2$	76.28	11.33	66.47	4.85	78.74	12.14	72.59	8.98
B4$_3$	76.21	9.62	66.67	4.70	78.92	10.44	72.88	7.04
B4$_4$	76.37	12.90	66.91	5.79	78.51	12.86	72.29	10.44
B6$_2$	77.06	14.61	67.02	5.65	79.03	11.65	73.42	8.98
B6$_3$	**82.69**	**28.01**	**75.15**	**11.66**	**83.37**	**18.69**	**79.8**	**15.29**
B6$_4$	76.95	14.46	66.77	5.21	79.16	12.14	73.39	8.98
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	80.79	43.46	76.58	26.52	72.44	27.43	70.91	23.30
R	83.06	15.23	78.66	10.82	83.40	17.48	81.72	14.56
B$_2$	91.06	53.30	87.60	39.05	92.91	37.86	91.33	33.01
B$_3$	90.59	49.58	86.65	35.05	92.85	40.05	91.19	34.95
B4$_2$	92.37	60.11	88.64	41.64	94.07	50.24	92.33	41.02
B4$_3$	92.60	59.34	88.88	42.77	**94.61**	52.43	**92.83**	43.20
B4$_4$	92.61	60.18	88.47	39.67	94.14	52.43	92.47	44.90
B6$_2$	92.60	63.13	88.82	42.99	94.23	52.43	92.59	42.72
B6$_3$	91.66	56.58	88.06	39.60	94.46	52.91	92.76	**45.39**
B6$_4$	**92.96**	**64.08**	**89.8**	**46.19**	94.24	**53.64**	92.76	44.66
Biaf	93.71	61.79	89.77	39.49	94.88	49.51	93.28	41.26
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	99.97	99.42	99.97	99.42	99.96	98.54	99.96	98.54
B$_3$	100	100	100	100	100	99.76	100	99.76
B4$_2$	99.87	97.45	99.87	97.45	99.67	87.14	99.67	87.14
B4$_3$	99.98	99.49	99.98	99.49	99.94	98.06	99.94	98.06
B4$_4$	100	100	100	100	99.98	99.27	99.98	99.27
B6$_2$	99.93	98.62	99.93	98.62	99.78	91.50	99.78	91.50
B6$_3$	99.99	99.71	99.99	99.71	99.95	98.30	99.95	98.30
B6$_4$	100	100	100	100	99.98	99.27	99.98	99.27

Table 15: Performance in the IWPT (French) dataset. Notation as in Table 8.

	dev				test			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	40.75	12.54	33.47	8.67	41.63	12.17	33.93	7.83
R	64.08	17.06	51.48	11.59	65.08	17.55	52.64	10.09
B$_2$	63.82	17.15	51.66	9.80	64.62	17.45	52.96	10.19
B$_3$	61.35	17.15	50.15	11.78	61.97	16.98	50.64	10.19
B4$_2$	65.88	16.31	**53.17**	9.52	65.55	15.66	53.65	8.49
B4$_3$	62.82	14.89	51.34	11.12	63.37	14.62	51.94	8.49
B4$_4$	64.07	15.55	51.59	9.43	64.99	15.75	52.89	8.58
B6$_2$	**66.38**	**19.6**	53.08	**12.25**	**66.25**	17.83	**53.87**	**11.04**
B6$_3$	65.67	17.53	52.55	11.03	66.03	17.45	53.34	10.28
B6$_4$	65.17	18.66	52.16	10.74	65.95	**18.21**	53.50	9.91
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	83.53	51.56	79.39	37.42	83.73	53.87	79.30	37.08
R	86.79	45.24	82.59	33.08	87.63	48.77	83.14	33.77
B$_2$	93.31	61.36	90.15	50.99	93.66	63.96	90.37	50.85
B$_3$	93.61	63.71	90.41	52.21	93.70	66.98	89.99	50.57
B4$_2$	93.79	64.66	90.41	51.84	94.45	67.36	91.09	52.55
B4$_3$	94.08	65.32	90.51	51.93	94.17	67.17	90.54	51.13
B4$_4$	93.85	63.90	90.32	49.67	94.14	66.70	90.33	50.38
B6$_2$	94.06	66.45	90.86	**53.35**	94.41	68.96	91.20	53.58
B6$_3$	**94.26**	**67.39**	**90.94**	53.25	**94.58**	**70.38**	**91.27**	**54.06**
B6$_4$	93.89	65.69	90.73	52.50	94.17	66.98	91.20	53.02
Biaf	94.22	54.19	90.61	41.38	94.25	54.25	90.10	37.92
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	99.79	96.89	99.79	96.89	99.88	97.83	99.88	97.83
B$_3$	99.96	99.72	99.96	99.72	100	99.91	100	99.91
B4$_2$	99.72	95.48	99.72	95.48	99.83	96.98	99.83	96.98
B4$_3$	99.94	99.25	99.94	99.25	99.99	99.72	99.99	99.72
B4$_4$	99.99	99.91	99.99	99.91	100	100	100	100
B6$_2$	99.85	97.64	99.85	97.64	99.92	98.40	99.92	98.40
B6$_3$	99.97	99.62	99.97	99.62	100	99.91	100	99.91
B6$_4$	99.99	99.91	99.99	99.91	100	100	100	100

Table 16: Performance in the IWPT (Slovak) dataset. Notation as in Table 8.

	dev				test			
BiLSTM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	32.19	0.00	23.37	0.00	32.79	0.00	23.54	0.00
R	60.54	0.83	41.71	0.00	60.99	0.00	44.96	0.00
B$_2$	62.62	4.17	42.91	0.00	65.52	**6.25**	47.71	2.50
B$_3$	62.62	4.17	42.91	0.00	65.52	6.25	47.71	2.50
B4$_2$	65.50	0.00	47.10	0.00	64.25	6.25	49.39	3.75
B4$_3$	65.66	3.33	48.34	0.00	62.96	5.00	**49.86**	**5.0**
B4$_4$	65.66	3.33	48.34	0.00	62.96	5.00	49.86	5.00
B6$_2$	**68.9**	**6.67**	**48.36**	**1.67**	**66.56**	6.25	49.51	3.75
B6$_3$	68.90	6.67	48.36	1.67	66.56	6.25	49.51	3.75
B6$_4$	68.90	6.67	48.36	1.67	66.56	6.25	49.51	3.75
XLM	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
A	34.22	0.00	27.27	0.00	35.78	3.75	29.67	2.50
R	64.69	1.67	53.69	0.00	63.82	3.75	53.29	0.00
B$_2$	73.62	11.67	62.02	**3.33**	75.24	**18.75**	64.18	**10.0**
B$_3$	73.62	11.67	62.02	3.33	75.24	18.75	64.18	10.00
B4$_2$	75.81	**16.67**	63.36	1.67	77.36	12.50	66.67	7.50
B4$_3$	76.01	11.67	**64.54**	2.50	**78.81**	15.00	**68.47**	10.00
B4$_4$	76.01	11.67	64.54	2.50	78.81	15.00	68.47	10.00
B6$_2$	**76.16**	15.83	63.48	2.50	76.66	15.00	66.18	7.50
B6$_3$	76.16	15.83	63.48	2.50	76.66	15.00	66.18	7.50
B6$_4$	76.16	15.83	63.48	2.50	76.66	15.00	66.18	7.50
Biaf	76.08	14.17	65.74	5.83	77.46	8.75	67.47	5.00
OC	**UF**	**UM**	**LF**	**LM**	**UF**	**UM**	**LF**	**LM**
B$_2$	100	100	100	100	100	100	100	100
B$_3$	100	100	100	100	100	100	100	100
B4$_2$	99.95	98.33	99.95	98.33	99.96	98.75	99.96	98.75
B4$_3$	100	100	100	100	100	100	100	100
B4$_4$	100	100	100	100	100	100	100	100
B6$_2$	100	100	100	100	100	100	100	100
B6$_3$	100	100	100	100	100	100	100	100
B6$_4$	100	100	100	100	100	100	100	100

Table 17: Performance in the IWPT (Tamil) dataset.

		tag-accuracy										%well-formed									
		A	**R**	**B_2**	**B_3**	**$B4_2$**	**$B4_3$**	**$B4_4$**	**$B6_2$**	**$B6_3$**	**$B6_4$**	**A**	**R**	**B_2**	**B_3**	**$B4_2$**	**$B4_3$**	**$B4_4$**	**$B6_2$**	**$B6_3$**	**$B6_4$**
		DAG																			
DM_{en}	**dev**	72.31	81.61	78.73	78.77	78.03	73.18	71.33	80.45	79.60	79.22	24.82	92.20	15.31	14.81	47.12	33.99	29.22	38.59	36.60	38.22
		92.02	92.45	88.36	88.45	93.84	92.28	91.40	93.97	93.63	93.64	79.61	86.46	78.67	81.19	94.73	87.50	88.76	87.71	81.74	87.30
		92.11	93.02	88.54	88.34	94.52	92.20	91.99	94.55	94.15	94.19	76.07	89.66	78.89	70.03	91.61	81.30	91.63	91.70	89.65	89.13
	id	71.97	81.07	77.61	77.84	77.22	72.29	70.32	79.59	78.91	78.78	22.70	91.88	14.92	14.67	47.97	34.49	30.69	36.79	36.38	41.47
		92.08	92.18	87.29	87.20	93.62	91.81	90.96	93.88	93.17	93.26	80.21	86.84	77.31	79.49	92.60	87.34	86.44	87.64	82.57	87.89
		92.46	93.05	87.60	87.36	93.99	91.93	91.68	94.20	94.08	93.97	76.88	87.14	76.04	66.87	92.44	78.55	87.72	90.61	87.31	85.96
	ood	68.60	76.80	76.98	77.16	70.01	65.07	63.11	75.24	74.29	74.75	29.03	96.12	13.66	12.07	46.95	35.36	27.92	36.46	33.60	38.04
		89.89	89.84	91.02	91.13	89.75	87.88	86.24	90.67	90.15	90.22	81.38	91.86	68.26	70.49	89.74	82.44	81.38	83.00	75.70	82.02
		90.31	90.88	91.69	91.25	90.35	87.65	87.89	91.34	91.04	91.18	80.30	92.19	68.06	58.90	88.62	70.35	82.19	78.33	82.65	81.94
PAS_{en}	**dev**	72.33	82.12	79.17	78.89	80.08	73.18	68.33	80.19	77.45	76.05	19.29	91.28	15.38	14.63	46.48	31.57	25.07	44.70	38.42	34.38
		90.89	91.20	88.40	88.40	94.19	91.87	90.25	93.43	92.58	92.13	62.28	76.58	81.68	82.74	93.98	90.21	88.34	92.57	90.86	92.37
		90.80	91.97	88.62	88.77	94.44	92.37	90.78	93.95	92.88	92.41	59.27	80.10	82.41	79.14	95.02	92.09	91.32	92.35	89.83	90.84
	id	72.05	81.73	77.02	76.67	79.37	72.06	67.37	79.69	77.05	75.76	18.76	90.49	12.55	13.03	47.49	28.91	24.37	43.02	36.04	31.46
		90.64	91.21	86.48	86.47	93.90	91.57	89.76	93.29	92.50	91.34	58.70	73.27	79.51	78.49	93.18	88.83	87.59	92.44	88.36	90.69
		90.88	91.67	86.58	86.52	94.51	91.95	90.65	93.42	92.69	92.16	58.24	77.53	80.50	73.74	94.79	90.14	90.54	90.83	87.30	88.68
	ood	68.23	77.64	71.44	70.92	72.28	64.86	60.42	74.61	71.52	70.24	24.04	92.06	14.23	12.61	41.86	29.00	26.43	43.27	34.37	33.39
		89.14	89.74	83.24	83.21	91.93	88.85	87.04	92.04	90.74	89.73	64.77	80.85	71.67	70.49	88.70	84.02	81.16	88.30	84.19	82.64
		89.84	91.03	83.82	83.56	92.52	89.45	88.34	92.37	91.23	90.75	67.30	83.96	73.75	67.15	92.00	85.70	84.14	85.43	82.11	83.36
PSD_{en}	**dev**	76.50	84.22	80.46	80.17	79.78	78.57	78.66	85.45	84.74	84.83	60.09	90.07	19.15	18.83	47.69	45.13	47.27	59.99	57.35	58.97
		93.14	92.42	88.23	88.01	92.15	91.61	91.80	93.91	93.67	93.68	90.88	90.07	72.36	69.10	86.23	87.51	87.83	88.43	85.27	85.65
		93.15	92.95	88.41	88.29	92.54	91.95	91.81	94.14	93.69	93.81	89.61	89.87	72.74	72.93	91.70	90.24	87.11	89.71	87.40	85.83
	id	76.48	83.86	79.02	78.93	79.01	77.79	78.36	84.50	84.12	84.24	59.88	90.83	18.14	17.46	47.49	45.68	46.68	57.34	56.33	56.50
		92.70	91.59	86.71	86.33	91.23	90.51	90.61	92.96	92.79	92.69	91.18	87.82	69.32	67.29	84.36	86.29	86.09	86.16	81.72	84.90
		92.55	92.36	86.65	86.52	91.56	90.96	90.93	93.14	92.75	92.93	89.79	87.58	67.66	71.29	89.24	86.42	88.46	87.32	84.22	85.88
	ood	76.62	81.94	77.64	77.07	75.78	75.14	75.37	83.08	82.60	82.45	69.54	94.38	22.37	22.20	50.03	47.44	47.95	60.52	59.55	59.11
		91.31	90.71	85.52	84.96	90.20	89.69	89.30	92.33	91.90	92.08	92.31	89.24	68.12	67.94	83.60	84.12	83.20	84.57	81.77	85.00
		92.08	91.68	85.72	85.64	90.44	89.91	89.65	92.74	92.27	92.14	91.10	88.96	66.61	67.26	86.72	86.08	85.27	87.04	83.12	84.25
PSD_{cs}	**dev**	67.76	80.67	76.35	76.59	74.45	73.30	73.92	81.77	81.48	81.60	35.13	91.78	12.26	12.97	41.40	42.21	47.06	54.10	56.63	55.07
		92.37	92.15	87.84	87.55	91.59	91.09	90.94	93.33	93.26	93.14	89.42	90.74	63.51	59.08	83.25	84.29	82.87	83.00	82.08	83.07
	id	67.31	79.25	74.92	75.24	73.39	72.10	72.86	80.98	80.53	80.87	35.45	89.90	11.77	12.60	42.26	41.42	46.21	58.68	54.94	55.79
		91.43	91.16	86.21	85.96	90.80	90.08	90.26	92.92	92.74	92.59	88.00	88.66	61.02	54.89	81.93	81.08	80.11	80.60	80.70	78.95
	ood	64.09	74.71	69.25	69.54	65.51	64.75	66.10	74.97	74.83	74.68	36.88	92.53	12.01	12.25	38.65	40.67	44.63	50.38	50.84	47.40
		89.00	88.65	83.81	83.70	87.47	86.42	86.21	89.98	89.55	89.41	88.39	86.20	55.13	52.41	78.52	76.87	75.53	79.72	77.46	77.01
PAS_{zh}	**dev**	53.07	68.76	68.89	68.87	58.37	53.49	50.27	63.98	60.08	58.78	6.59	93.26	5.54	5.18	18.89	12.55	9.62	18.47	12.22	13.21
		81.67	81.98	81.37	81.52	82.43	79.75	78.17	84.60	82.60	81.69	39.52	39.83	39.89	42.07	61.91	60.48	56.69	62.91	57.46	58.08
	id	55.30	70.25	70.17	69.78	61.40	56.25	52.49	66.09	62.03	60.67	6.10	90.74	5.33	5.18	23.79	14.40	9.77	22.11	13.51	14.21
		82.67	82.47	81.06	81.06	83.04	79.96	78.56	84.51	82.93	81.95	43.84	40.50	42.93	44.87	68.09	65.52	60.79	66.18	63.11	63.32
μ	**dev**	68.39	79.48	76.72	76.66	74.14	70.34	68.50	78.37	76.67	76.10	29.18	91.72	13.53	13.28	40.32	33.09	31.65	43.17	40.24	39.97
		90.02	90.04	86.84	86.79	90.84	89.32	88.51	91.85	91.15	90.86	72.34	76.74	67.22	66.84	84.02	82.00	80.90	82.92	79.48	81.29
		92.02	92.65	88.52	88.47	93.83	92.17	91.53	94.21	93.57	93.47	74.98	86.54	78.01	74.03	92.78	87.88	90.02	91.25	88.96	88.60
	id	68.62	79.23	75.75	75.69	74.08	70.10	68.28	78.17	76.53	76.06	28.58	90.77	12.54	12.59	41.80	32.98	31.54	43.59	39.44	39.89
		89.90	89.72	85.55	85.40	90.52	88.79	88.03	91.51	90.83	90.37	72.39	75.42	66.02	65.01	84.03	81.81	80.20	82.60	79.29	81.15
		91.96	92.36	86.94	86.80	93.35	91.61	91.09	93.59	93.17	93.02	74.97	84.08	74.73	70.63	92.16	85.04	88.91	89.59	86.28	86.84
	ood	69.47	77.80	72.69	72.50	70.95	67.51	66.33	77.02	75.86	75.57	40.95	93.78	15.99	15.31	44.88	38.65	37.34	48.24	45.08	45.02
		89.86	89.76	83.89	83.78	89.87	88.24	87.24	91.28	90.60	90.38	82.12	87.12	66.46	65.84	85.38	82.17	80.60	84.17	80.16	81.94
		90.78	91.22	84.38	84.18	91.13	89.04	88.66	92.10	91.53	91.38	80.29	88.65	70.21	65.10	89.41	81.38	84.23	86.35	82.94	83.62
		IWPT																			
ar	**dev**	48.45	70.79	72.86	72.51	68.58	69.23	69.05	62.67	72.16	72.26	12.45	63.97	4.44	6.36	47.94	58.28	46.98	43.74	45.66	56.05
		79.67	81.26	83.62	83.42	83.72	83.68	83.25	84.65	84.62	84.77	59.45	67.84	27.58	26.75	64.79	68.80	70.85	58.22	69.71	68.86
	test	43.87	71.93	73.86	73.65	69.11	69.86	70.11	62.89	73.28	73.68	10.66	76.36	4.54	5.70	45.62	55.65	45.64	43.89	44.01	57.10
		73.72	82.27	85.12	85.00	84.80	84.54	84.40	85.99	86.07	86.21	47.37	62.33	23.73	23.08	56.29	57.83	62.60	52.21	62.60	58.19
fi	**dev**	38.21	60.40	60.51	58.98	55.28	56.41	54.78	63.32	62.29	62.73	30.83	98.73	8.70	9.66	33.83	33.71	40.17	32.55	39.34	36.54
		81.30	85.17	81.57	81.17	89.14	88.82	88.41	90.66	90.53	90.49	76.71	91.35	52.49	48.50	78.91	77.61	74.84	79.86	76.25	78.75
	test	37.20	60.54	61.38	59.29	54.70	56.51	55.09	62.97	62.29	63.30	24.52	98.36	5.70	6.14	26.77	28.25	32.10	30.22	35.64	35.32
		82.77	84.55	81.34	80.95	86.39	85.86	85.77	89.12	88.73	88.73	78.06	93.33	50.83	48.66	82.75	80.53	79.62	82.04	79.31	79.09
fr	**dev**	40.13	73.62	77.75	77.73	74.22	74.13	73.34	78.23	81.90	78.61	17.77	72.92	7.63	9.22	51.89	48.71	47.83	49.06	50.39	56.73
		70.70	83.06	88.97	89.06	92.79	92.66	92.02	93.80	93.74	93.39	49.64	81.09	35.68	34.55	73.65	75.51	78.76	78.83	78.08	74.59
	test	52.48	72.18	76.38	77.30	74.13	74.16	74.55	77.10	82.25	77.42	36.39	73.78	14.18	12.84	46.52	46.57	53.89	39.26	59.61	51.03
		79.37	81.70	89.43	88.79	91.63	91.09	91.56	91.91	91.06	92.49	66.69	69.49	48.87	47.58	80.18	81.57	81.95	84.51	82.01	80.02
sk	**dev**	37.97	63.80	64.86	64.31	63.12	62.38	62.83	68.09	68.29	67.98	21.29	98.54	12.34	11.49	46.36	45.96	54.44	41.53	35.69	37.15
		82.57	87.39	88.30	88.23	93.13	92.47	92.58	94.15	93.96	93.67	72.18	92.09	58.95	59.39	84.82	80.58	81.70	84.98	82.66	85.06
	test	37.11	62.66	63.34	63.16	62.34	60.24	60.95	67.31	67.62	67.59	20.45	98.31	10.89	9.67	49.16	47.54	58.54	45.21	40.34	38.75
		82.18	86.35	86.74	86.77	91.88	91.82	91.28	93.38	93.18	92.94	75.77	90.35	59.37	59.31	84.63	82.47	82.89	84.86	83.89	85.85
ta	**dev**	28.59	58.60	64.85	64.85	58.44	56.93	56.93	65.73	65.73	65.73	9.19	46.07	14.57	14.57	42.73	42.98	42.98	52.35	52.35	52.35
		33.70	62.76	74.20	74.20	73.73	74.28	74.28	77.12	77.12	77.12	18.92	61.75	15.49	15.49	55.41	53.75	53.75	69.63	69.63	69.63
	test	28.10	60.68	63.61	63.61	59.03	58.83	58.83	65.00	65.00	65.00	7.75	43.05	7.19	7.19	37.44	59.33	59.33	42.48	42.48	42.48
		27.73	64.00	76.38	76.38	72.67	73.50	73.50	77.09	77.09	77.09	11.37	60.67	36.17	36.17	68.32	63.27	63.27	72.41	72.41	72.41
μ	**dev**	30.07	60.55	62.16	58.30	56.77	55.93	55.04	61.32	60.61	61.25	17.75	99.31	4.96	5.61	31.97	41.31	44.45	38.51	34.06	37.80
		73.15	82.34	83.76	82.89	88.12	87.17	87.25	89.83	89.58	89.16	60.34	81.63	38.39	34.87	74.71	68.54	69.32	72.99	74.92	69.39
	test	28.12	59.27	62.38	57.94	55.84	55.57	54.86	60.76	60.78	60.05	15.52	99.50	4.53	5.26	30.65	35.80	40.05	34.48	28.77	34.40
		67.84	81.27	84.56	83.88	87.40	87.05	86.72	89.39	88.99	88.89	54.65	79.55	33.09	29.94	67.23	65.46	64.19	65.81	63.31	61.53

Table 18: Performance in terms of tagging accuracy (**tag-accuracy**) and percentage of well-formed trees without post-processing (**%well-formed**) for the DAG and IWPT treebanks. Each subrow determines the encoder used (BiLSTM, XLM and, for treebanks in English, XLNet). The average performance is also included (μ).

			BiLSTM										XLM | XLNet										
		k	A	R	B_2	B_3	$B4_2$	$B4_3$	$B4_4$	$B6_2$	$B6_3$	$B6_4$	A	R	B_2	B_3	$B4_2$	$B4_3$	$B4_4$	$B6_2$	$B6_3$	$B6_4$	Biaf
DAG																							
DM_{en}	id	1	61.63	79.89	82.72	83.29	74.07	80.54	82.18	78.00	81.69	82.14	89.73	92.52	95.44	95.14	85.95	93.02	95.36	91.45	95.34	95.66	95.32
		2	57.40	77.91	79.19	79.74	72.49	77.79	79.54	75.81	78.66	79.57	87.92	91.59	95.28	94.51	87.09	92.48	95.02	91.64	94.75	95.03	95.01
		3	53.44	70.83	72.32	72.37	66.47	72.76	72.71	67.31	70.02	73.09	81.97	87.38	87.91	87.17	84.31	88.67	90.73	88.05	89.47	91.21	92.21
		4	25.81	44.44	56.25	52.94	48.48	38.71	55.17	48.48	45.71	43.75	72.73	57.14	57.14	54.55	58.82	64.71	64.71	58.82	62.86	62.86	64.71
	ood	1	61.31	75.28	77.48	77.08	70.44	74.93	76.59	73.62	76.47	77.14	88.71	90.60	92.71	92.37	85.15	90.23	92.56	89.16	92.53	93.03	92.51
		2	54.61	73.23	74.38	74.26	69.14	73.29	74.17	71.29	73.68	74.62	86.63	89.59	92.66	91.73	86.15	89.95	92.48	89.22	91.76	92.58	92.75
		3	43.81	67.28	66.05	65.75	63.75	64.92	66.77	60.95	64.92	66.36	72.75	79.10	83.98	82.61	78.73	81.86	84.57	81.05	85.43	85.76	85.67
PAS_{en}	id	1	68.16	83.13	86.03	85.51	71.39	80.27	83.12	78.68	83.55	84.77	88.30	91.82	95.96	95.93	80.53	90.26	94.27	88.70	94.16	95.69	95.76
		2	63.37	79.63	82.98	82.28	68.50	77.17	79.29	75.52	80.46	82.48	85.31	89.03	95.71	95.58	79.27	89.15	93.42	86.91	93.10	94.99	95.63
		3	59.22	74.06	78.31	80.21	65.77	73.81	73.01	73.44	77.00	79.39	85.40	88.85	95.36	95.06	79.10	88.59	91.80	84.69	92.93	96.19	96.23
	ood	1	67.59	79.91	82.74	82.32	71.17	77.76	79.68	77.77	81.39	81.81	88.99	91.59	94.94	94.85	81.86	90.28	93.43	89.11	93.67	94.41	94.15
		2	59.55	75.73	78.49	77.80	67.09	73.45	75.35	73.22	77.05	77.82	84.05	89.07	94.83	94.82	81.02	89.36	92.80	87.75	93.02	94.36	94.22
		3	46.32	65.51	70.14	66.67	58.89	66.67	62.72	63.27	65.60	67.59	80.66	83.55	89.14	87.70	79.05	83.82	87.64	82.70	89.04	89.50	88.50
PSD_{en}	id	1	71.97	81.27	81.90	82.02	82.50	81.73	82.30	84.13	83.37	83.83	92.67	92.02	93.92	93.66	94.29	93.85	93.92	93.91	93.58	93.68	93.80
		2	61.75	76.64	77.77	77.66	79.07	77.51	78.53	79.75	80.44	80.00	88.24	88.98	91.82	91.95	92.38	92.33	92.36	92.18	92.31	92.40	92.55
		3	52.79	70.26	72.76	71.56	71.85	71.72	72.45	73.07	74.34	74.59	81.93	83.34	87.54	88.29	88.62	88.82	89.73	88.85	89.53	89.74	90.68
		4	48.02	65.98	70.79	69.73	67.62	65.14	67.09	70.25	69.96	68.18	77.48	77.54	83.46	84.99	87.40	88.89	88.93	90.66	89.39	91.81	92.34
	ood	1	73.61	79.41	80.12	80.27	80.47	79.85	80.69	82.05	81.94	81.96	92.73	92.22	93.23	93.23	93.13	93.40	93.18	93.31	93.48	93.14	93.10
		2	59.26	72.41	74.28	73.59	73.88	73.93	74.08	75.12	75.26	75.93	86.51	87.04	90.69	90.82	90.50	91.08	90.69	91.06	90.79	91.08	90.63
		3	51.00	65.82	69.68	69.80	67.79	68.86	68.40	70.83	71.88	70.59	81.08	80.65	88.32	89.11	88.89	89.96	90.29	89.09	90.26	91.15	91.22
		4	46.22	57.06	64.11	65.51	62.78	64.98	63.16	66.76	66.38	64.25	71.00	71.15	81.47	82.12	83.98	85.41	86.20	85.11	84.60	85.31	87.25
		5	29.41	49.06	52.63	51.55	55.45	51.92	53.85	60.78	64.71	62.00	65.45	67.29	71.70	78.90	77.78	64.81	80.36	79.25	78.18	84.68	88.14
PSD_{cs}	id	1	64.17	78.73	78.86	79.44	80.39	80.26	79.82	81.85	81.79	82.15	93.60	92.95	94.77	94.78	94.70	94.72	94.86	94.92	95.20	94.79	94.97
		2	52.42	72.10	73.30	73.80	74.51	73.62	74.05	75.88	75.95	76.54	88.91	88.12	91.97	92.11	92.38	92.42	92.69	92.73	92.84	92.65	93.31
		3	43.18	65.14	67.92	67.79	69.01	69.00	68.71	71.40	72.96	70.76	81.06	82.79	88.21	87.08	87.56	89.85	90.61	90.26	88.61	90.39	90.02
		4	34.15	64.70	63.27	68.70	71.82	66.10	63.05	68.23	67.01	69.73	68.57	76.85	82.16	82.72	83.87	85.09	88.01	84.68	88.10	86.44	84.99
	ood	1	61.90	73.75	72.91	73.59	73.75	73.21	74.09	75.15	75.36	75.40	91.59	90.78	92.75	93.13	93.17	92.66	93.10	93.21	93.30	93.05	93.17
		2	48.12	64.99	64.89	65.76	65.55	65.10	65.47	67.91	68.07	67.53	84.90	84.94	88.64	88.64	89.08	88.81	88.88	89.21	89.31	89.18	89.38
		3	38.28	55.81	57.18	58.70	56.78	56.19	56.88	59.52	59.43	59.35	75.70	75.19	81.78	82.24	81.97	83.48	82.95	84.18	83.11	83.71	84.79
		4	35.10	54.38	56.60	56.34	55.18	54.26	56.06	59.87	58.82	60.72	73.52	72.54	78.61	77.72	79.68	81.87	80.82	80.50	80.48	82.44	83.99
		5	35.09	52.13	54.29	50.00	50.57	50.10	48.85	53.74	58.51	57.25	65.36	64.31	74.67	77.50	76.28	80.29	79.93	80.94	79.42	81.71	86.31
		6	33.33	36.07	24.49	39.22	38.60	23.08	30.77	49.18	54.24	50.79	64.52	61.54	51.72	55.74	64.41	75.00	63.16	71.88	68.85	68.85	91.43
PAS_{zh}	id	1	45.43	69.50	73.18	73.19	62.69	67.80	68.66	66.92	69.84	71.25	79.59	80.71	89.13	89.11	77.71	84.26	86.93	82.48	86.48	87.90	88.32
		2	37.09	65.81	68.43	68.19	59.81	63.88	64.11	63.14	65.16	66.51	73.30	77.29	86.57	86.64	77.47	83.27	85.60	81.49	85.11	86.36	87.40
		3	30.46	61.71	63.72	63.38	55.07	58.49	59.43	59.83	60.61	61.81	64.81	73.59	82.30	82.98	75.18	80.99	82.68	79.07	82.07	84.10	85.11
μ	id	1	62.27	78.50	80.54	80.69	74.21	78.12	79.21	77.92	80.05	80.83	88.78	90.00	93.85	93.73	86.64	91.22	93.07	90.29	92.95	93.54	93.63
		2	54.40	74.42	76.33	76.34	70.88	74.00	75.11	74.02	76.13	77.02	84.74	87.00	92.27	92.16	85.72	89.93	91.82	88.99	91.62	92.28	92.78
		3	47.82	68.40	71.00	71.06	65.63	69.16	69.26	69.01	70.99	71.93	79.04	83.19	88.27	88.12	82.95	87.38	89.11	86.19	88.52	90.33	90.85
		4	35.99	58.37	63.44	63.79	62.64	56.65	61.77	62.32	60.90	60.55	72.93	70.51	74.26	74.09	76.70	79.56	80.55	78.05	80.12	80.37	80.68
	ood	1	66.10	77.09	78.31	78.32	73.96	76.44	77.76	77.15	78.79	79.08	90.51	91.30	93.41	93.40	88.33	91.64	93.07	91.19	93.24	93.41	93.23
		2	55.38	71.59	73.01	72.85	68.91	71.44	72.27	71.89	73.51	73.97	85.52	87.66	91.71	91.50	86.69	89.80	91.21	89.31	91.22	91.80	91.74
		3	44.86	63.60	65.76	65.23	61.80	64.16	63.69	63.64	65.46	65.97	77.55	79.62	85.81	85.42	82.16	84.78	86.36	84.25	86.96	87.53	87.55
		4	40.66	55.72	60.35	60.93	58.98	59.62	59.61	63.32	62.60	62.48	72.26	71.85	80.04	79.92	81.83	83.64	83.51	82.80	82.54	83.87	85.62
		5	32.25	50.59	53.46	50.77	53.01	51.01	51.35	57.26	61.61	59.63	65.41	65.80	73.18	78.20	77.03	72.55	80.14	80.09	78.80	83.19	87.22
IWPT																							
ar		1	54.33	74.80	77.06	78.37	77.19	78.62	78.33	69.55	78.46	78.99	83.44	84.99	90.29	90.26	90.63	90.30	90.19	90.31	90.38	90.85	92.11
		2	41.44	70.25	73.35	73.72	73.00	74.04	73.55	66.52	73.85	74.45	69.43	80.50	86.59	86.69	86.49	86.81	87.18	86.48	87.27	87.17	88.66
		3	39.80	68.87	72.98	73.75	72.27	73.84	73.63	64.24	74.10	74.29	66.34	78.65	85.46	84.82	85.15	83.93	86.35	85.12	85.40	85.06	87.23
		4	41.85	60.02	61.99	63.43	63.14	64.58	62.43	58.07	66.60	64.83	55.25	69.93	74.93	74.58	71.46	73.63	73.86	70.82	72.01	74.46	75.44
		5	24.58	46.70	39.05	40.45	42.39	40.45	38.86	33.12	38.97	39.77	28.57	53.33	60.11	56.35	55.10	56.38	51.06	56.70	50.00	59.46	55.03
fi		1	43.93	63.63	59.96	61.37	57.95	59.67	58.89	60.73	61.23	61.73	87.58	87.70	92.13	92.18	92.24	92.06	92.02	92.47	92.05	92.45	93.44
		2	35.26	62.07	59.18	59.74	56.33	58.16	57.46	60.14	58.92	57.96	81.43	84.27	90.75	90.57	91.46	91.27	91.50	91.85	91.54	91.76	93.46
		3	33.30	59.11	55.69	57.61	55.09	56.40	55.46	60.10	58.14	57.42	78.65	81.79	87.91	88.08	90.41	90.39	90.10	90.43	89.88	90.52	94.47
		4	25.11	52.39	46.08	50.15	45.75	49.79	48.86	52.54	55.19	51.00	68.26	78.24	82.91	85.57	83.71	88.17	91.46	87.27	90.75	91.78	94.12
		5	17.86	42.62	23.53	26.92	34.48	32.00	43.64	43.64	36.36	45.28	73.33	61.29	71.43	72.41	76.67	74.58	78.69	75.86	64.41	77.19	93.75
		6	17.14	40.00	31.75	37.50	31.88	37.14	28.99	45.07	42.67	42.67	52.87	55.81	61.11	68.42	60.00	65.85	62.50	78.48	66.67	86.05	96.84
fr		1	59.68	74.40	77.41	77.73	76.75	76.54	76.92	77.68	83.78	77.48	84.12	83.63	91.67	91.06	92.78	93.03	93.16	93.10	91.98	93.68	94.02
		2	39.64	72.15	73.60	73.29	74.53	75.09	74.60	74.71	78.58	74.86	69.42	81.18	88.88	89.06	90.79	91.07	90.70	90.83	90.47	90.27	93.02
		3	28.59	70.95	73.13	72.06	73.61	73.87	70.88	74.24	77.22	75.08	52.69	78.09	86.96	85.85	90.79	89.66	88.64	89.30	90.32	89.59	87.90
sk		1	46.49	65.75	65.67	63.38	67.42	65.21	66.27	67.47	66.15	65.97	89.41	90.21	94.51	94.73	94.38	94.93	94.71	94.61	94.83	94.51	94.43
		2	35.61	62.97	62.60	60.09	64.95	60.81	62.38	65.87	65.69	64.86	78.73	84.24	92.87	93.06	93.63	93.83	93.55	94.00	94.20	93.69	94.27
		3	32.74	58.30	56.95	53.06	60.78	58.76	58.97	61.33	62.90	61.56	74.96	78.64	87.66	88.85	90.84	90.09	89.94	90.37	90.46	91.49	93.16
		4	38.96	64.20	72.97	74.67	61.11	65.71	61.33	72.00	64.20	65.82	76.54	82.35	80.00	94.12	92.86	85.00	91.57	89.16	91.57	83.54	92.86
		5	22.45	58.82	52.75	43.18	47.92	43.18	47.42	52.08	40.40	47.31	29.13	71.15	74.75	76.77	75.51	76.92	74.51	84.00	80.81	78.00	77.78
ta		1	34.90	61.82	64.83	64.83	67.45	67.89	67.89	71.76	71.76	71.76	36.84	66.37	74.57	74.57	76.54	77.23	77.23	77.65	77.65	77.65	78.49
		2	17.08	53.20	49.54	49.54	53.79	52.52	52.52	52.84	52.84	52.84	19.83	55.21	68.06	68.06	71.60	69.04	69.04	67.56	67.56	67.56	63.43
μ		1	47.86	68.08	68.99	69.14	69.35	69.59	69.66	69.44	72.28	71.19	76.28	82.58	88.63	88.56	89.31	89.51	89.46	89.63	89.38	89.83	90.50
		2	33.81	64.13	63.65	63.28	64.52	64.12	64.10	64.01	65.98	64.99	63.77	77.08	85.43	85.49	86.79	86.40	86.39	86.14	86.21	86.09	86.57
		3	33.61	64.31	64.69	64.12	65.44	65.72	64.74	64.98	68.09	67.09	68.16	79.29	87.00	86.90	89.30	88.52	88.76	88.80	89.01	89.17	90.69
		4	35.31	58.87	60.35	62.75	56.66	60.03	57.54	60.87	62.00	60.55	66.69	76.84	79.28	84.75	82.67	82.27	85.63	82.41	84.77	83.26	87.47
		5	21.63	49.38	38.44	36.85	41.60	38.54	43.31	42.95	38.58	44.12	43.68	61.93	68.76	68.51	69.09	69.29	68.09	72.19	65.07	71.55	75.52

Table 19: UF score per plane in the DAG and IWPT evaluation treebanks. Each subrow indicates the score achieved for graphs with k planes and the average performance across treebanks is included (μ).

		BiLSTM										XLM										
	m	**A**	**R**	**B$_2$**	**B$_3$**	**B4$_2$**	**B4$_3$**	**B4$_4$**	**B6$_2$**	**B6$_3$**	**B6$_4$**	**A**	**R**	**B$_2$**	**B$_3$**	**B4$_2$**	**B4$_3$**	**B4$_4$**	**B6$_2$**	**B6$_3$**	**B6$_4$**	**Biaf**
ar	0	51.23	72.10	74.56	75.53	74.51	75.84	75.53	67.66	75.95	76.21	78.21	82.32	87.67	87.63	87.65	87.67	87.89	87.74	88.11	88.27	89.53
	1	43.48	71.99	75.41	76.10	74.98	76.19	75.63	67.53	75.59	76.63	75.26	82.08	88.54	88.48	88.55	88.59	88.69	88.11	88.40	88.94	90.68
	2	38.03	71.38	74.54	75.27	73.88	74.87	74.58	68.44	75.00	75.23	62.28	81.59	88.06	87.73	88.16	87.09	87.67	86.69	87.72	87.94	88.77
	3	30.52	69.90	71.93	71.09	72.43	72.42	72.13	64.79	72.77	74.39	56.79	81.28	85.88	87.04	86.06	85.79	87.99	85.92	86.75	86.51	88.82
	4	25.86	68.63	71.30	72.94	73.56	76.00	73.85	60.65	74.23	73.58	47.50	76.41	85.16	85.14	84.63	86.42	86.70	85.90	86.81	85.11	86.20
fi	0	41.77	62.84	59.33	60.54	57.31	59.08	58.19	60.52	60.84	61.23	85.68	86.42	91.57	91.52	91.81	91.79	91.68	92.17	91.75	92.09	93.58
	1	34.21	61.73	58.73	59.87	55.55	57.46	57.03	59.45	57.44	55.76	82.77	85.28	90.11	90.42	91.54	90.91	91.48	91.37	91.22	91.70	93.11
	2	31.06	62.66	57.16	59.55	57.96	59.26	59.27	61.22	58.84	54.53	71.30	81.52	92.98	91.69	91.78	92.37	92.83	92.20	92.02	92.87	94.34
	3	34.05	62.90	58.21	57.47	58.95	58.70	57.61	61.39	58.28	57.83	78.39	81.95	89.12	89.61	93.09	91.58	92.23	93.64	92.86	94.03	95.40
	4	42.42	66.67	53.33	50.85	41.38	53.33	43.75	50.79	53.33	45.45	52.94	65.71	66.67	77.61	86.57	75.36	76.47	80.56	76.47	68.66	71.43
	5	23.78	51.55	52.87	48.31	39.13	48.68	46.67	51.02	42.16	41.05	72.73	72.20	82.29	81.44	72.92	83.51	88.44	86.73	86.70	94.17	94.39
	6	35.90	55.00	75.00	71.43	59.09	53.33	55.81	66.67	73.17	50.00	73.91	77.27	88.37	88.37	88.37	90.48	93.02	90.91	90.91	84.44	95.45
	7	20.62	41.90	41.76	55.32	42.42	41.18	40.82	56.31	57.14	56.60	48.60	65.52	68.63	71.84	81.13	82.88	85.45	89.09	92.17	85.45	90.77
	8	60.47	79.55	85.06	80.49	80.46	81.40	89.66	85.71	84.09	85.06	92.63	87.91	97.87	97.87	94.51	95.65	96.77	96.77	97.87	97.87	97.87
	11	46.28	74.60	74.78	82.05	70.59	69.57	76.52	71.79	72.27	75.21	65.62	81.20	91.60	92.31	88.37	91.47	95.45	95.45	90.77	88.72	95.52
	13	24.32	43.04	46.38	46.88	52.94	39.47	35.14	53.33	38.36	59.46	72.29	74.70	61.76	62.86	82.05	85.71	78.05	86.42	89.16	90.48	98.90
	15	59.26	71.56	77.36	73.58	71.56	76.64	81.08	74.77	75.47	70.37	79.66	81.03	91.38	91.67	93.10	90.43	90.60	89.08	87.18	92.44	98.39
fr	0	57.45	73.97	76.57	76.94	76.29	76.16	76.48	77.27	83.06	77.00	82.38	83.05	90.91	90.45	92.32	92.53	92.58	92.60	91.55	92.86	93.73
	1	45.52	74.09	77.20	76.37	76.29	76.59	76.04	76.00	81.30	76.62	73.28	83.56	92.14	91.42	92.58	93.02	93.39	92.91	92.53	93.37	93.74
	2	35.06	72.25	75.68	76.11	76.70	76.11	74.87	76.26	79.10	78.05	65.62	82.01	92.62	92.29	93.52	94.17	92.60	93.17	92.19	95.16	93.52
	3	34.10	71.01	76.39	70.43	74.01	77.37	74.96	72.65	73.97	72.67	61.81	80.86	88.86	88.99	90.83	90.28	87.08	87.43	90.78	91.35	92.46
sk	0	43.80	64.23	64.88	62.93	66.74	64.43	65.54	67.52	66.12	65.80	86.67	87.84	93.81	94.24	94.05	94.52	94.18	94.47	94.75	94.21	94.60
	1	27.91	62.17	58.39	54.36	61.34	54.75	57.34	61.38	62.59	61.93	70.51	82.35	91.49	91.60	92.90	92.18	92.79	92.55	92.55	92.69	92.41
	2	32.25	65.70	59.43	54.38	63.70	60.32	61.22	61.36	66.89	62.88	70.74	82.97	89.46	87.50	92.09	91.26	90.51	88.71	89.27	90.54	90.77
	3	28.88	71.18	67.72	59.66	67.20	65.05	62.30	68.54	70.23	68.72	75.18	84.82	92.72	91.40	92.94	95.22	93.98	96.42	95.69	94.94	97.42
	4	22.90	67.61	72.73	60.15	73.91	64.71	61.19	61.31	70.92	64.23	77.14	86.30	93.15	91.03	91.03	93.88	89.66	95.24	93.88	93.88	95.95
	7	27.12	70.00	50.00	56.60	61.82	49.12	46.43	56.14	61.02	66.67	55.74	75.00	84.75	91.80	92.06	91.80	95.24	90.00	83.33	90.00	95.65
ta	0	34.55	62.48	65.04	65.04	68.38	68.41	68.41	72.02	72.02	72.02	36.93	65.55	76.58	76.58	78.06	77.35	77.35	78.59	78.59	78.59	78.87
	1	29.80	57.43	58.43	58.43	61.57	61.32	61.32	63.94	63.94	63.94	32.38	63.58	69.71	69.71	72.56	73.65	73.65	71.85	71.85	71.85	72.58
	2	22.22	61.36	61.45	61.45	61.54	63.03	63.03	69.05	69.05	69.05	17.54	61.71	64.80	64.80	71.51	76.14	76.14	79.33	79.33	79.33	65.93
	3	19.05	51.06	68.09	68.09	46.15	63.41	63.41	59.57	59.57	59.57	14.63	65.22	72.73	72.73	74.42	78.05	78.05	68.09	68.09	68.09	79.17
μ	0	45.76	67.13	68.08	68.19	68.65	68.78	68.83	69.00	71.60	70.45	73.97	81.04	88.11	88.08	88.78	88.77	88.74	89.12	88.95	89.20	90.06
	1	36.18	65.48	65.63	65.03	65.95	65.26	65.47	65.66	68.17	66.98	66.84	79.37	86.40	86.33	87.63	87.67	88.00	87.36	87.31	87.71	88.50
	2	31.72	66.67	65.65	65.35	66.76	66.72	66.59	67.27	69.77	67.95	57.50	77.96	85.58	84.80	87.41	88.20	87.95	88.02	88.10	89.17	86.67
	3	29.32	65.21	68.47	65.35	63.75	67.39	66.08	65.39	66.97	66.64	57.36	78.82	85.86	85.95	87.47	88.18	87.87	86.30	86.83	86.99	90.65
	4	30.40	67.63	65.79	61.31	62.95	64.68	59.60	57.59	66.16	61.09	59.19	76.14	81.66	84.60	87.41	85.22	84.28	87.23	85.72	82.55	84.53
	7	23.87	55.95	45.88	55.96	52.12	45.15	43.62	56.23	59.08	61.64	52.17	70.26	76.69	81.82	86.60	87.34	90.35	89.55	87.75	87.73	93.21

Table 20: UF score per number of cycles in the IWPT evaluation treebanks. Each subrow indicates the score achieved for graphs with m cycles and the last row shows the average across treebanks (μ).