Word Reordering for Zero-shot Cross-lingual Structured Prediction

Tao Ji[*], **Yong Jiang**, **Tao Wang**, **Zhongqiang Huang**,
Fei Huang, **Yuanbin Wu**, **Xiaoling Wang**
School of Computer Science and Technology,
East China Normal University
DAMO Academy, Alibaba Group
{taoji@stu,ybwu@cs,xlwang@cs}.ecnu.edu.cn
{yongjiang.jy,leeo.wangt,z.huang,f.huang}@alibaba-inc.com

Abstract

Adapting word order from one language to another is a key problem in cross-lingual structured prediction. Current sentence encoders (e.g., RNN, Transformer with position embeddings) are usually word order sensitive. Even with uniform word form representations (MUSE, mBERT), word order discrepancies may hurt the adaptation of models. This paper builds structured prediction models with bag-of-words inputs. It introduces a new reordering module to organize words following the source language order, which learns task-specific reordering strategies from a general-purpose order predictor model. Experiments on zero-shot cross-lingual dependency parsing, POS tagging, and morphological tagging show that our model can significantly improve target language performances, especially for languages that are distant from the source language. [1]

1 Introduction

Extracting linguistic structures from natural language usually relies on high quality human annotations. To handle low resource scenarios, efforts have been devoted to sharing resources among languages and adapting from high resource models. One crucial step of these methods is how to unify input and output spaces over languages. For example, the universal dependency project (McDonald et al., 2013) constructs a universal output space for cross-lingual dependency parsers, and cross-lingual word representation learning algorithms helps aligning word forms of different languages (Conneau et al., 2017; Devlin et al., 2019).

Beyond word form, word order is another important factor in cross-lingual structured prediction (Wang and Eisner, 2018b): it is possible that sentences in two different language have similar parse trees, but their words are organized in different orders (e.g., SVO or SOV). To share annotations among them, we need to handle word order discrepancies carefully: if a model learned on the source language is tightly coupled with the source language word order, performances on target languages could be hurt as their word order could be incompatible (Wang et al., 2019). On the other side, if one completely drops word order (e.g., bag-of-words), the source language (and target languages) performances might be poor as order-sensitive features could be essential. Trade-offs have been made by using weak word order information (e.g., relative positions instead of absolute positions (Ahmad et al., 2019a)), but we still want to seek better adaptation of word order without sacrificing source language performances.

In this work, we integrate new *reordering* modules to help cross-lingual structured prediction. Given a bag-of-words from the target language, the module tries to reorder them to best resemble a source language sentence. The structured prediction part then receives inputs with a more familiar order information. Crucially,

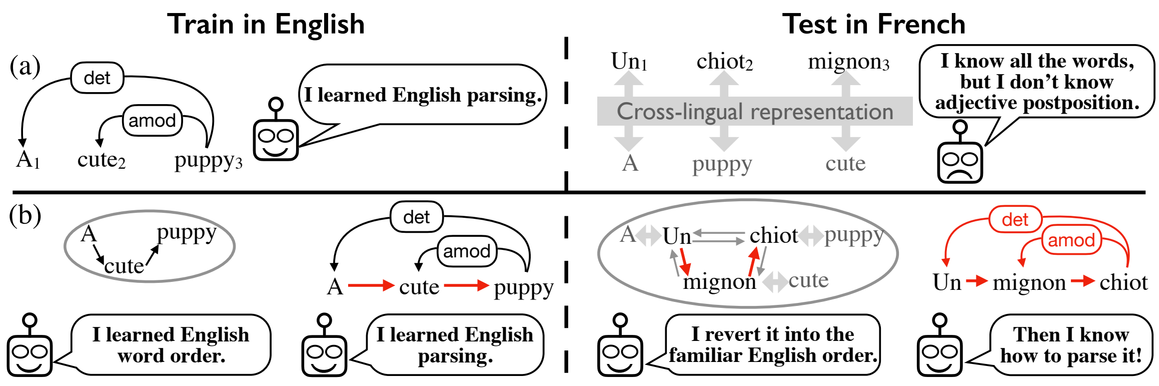
- the training of the reordering model only requires unlabelled source language data, without parallel corpora or off-the-shelf word alignment tools (Tiedemann et al., 2014; Zhang et al., 2019) (thanks to the universal word forms).
- we don't really need to perform the reordering action. Instead, the correct order can be implicitly encoded by multi-task learning: word order information accesses the model as a supervision signal.

The separation of reordering module and structured prediction module provides a new way to both explore and transfer order information.

We suggest a distillation framework (Hinton et al., 2015) for learning the reordering module.

[*]This work was conducted when Tao Ji was interning at Alibaba DAMO Academy.

[1]https://github.com/AntNLP/zero-shot-structured-prediction.



Train in English
Test in French
(a)
det
amod
A1 cute2 puppy3
I learned English parsing.
Un1 chiot2 mignon3
Cross-lingual representation
A puppy cute
I know all the words, but I don't know adjective postposition.
(b)
A puppy cute
I learned English word order.
det
amod
A cute puppy
I learned English parsing.
A Un chiot puppy mignon cute
I revert it into the familiar English order.
det
amod
Un mignon chiot
Then I know how to parse it!

Figure 1: A diagram of the traditional zero-shot cross-lingual transfer approach (a) and the reordering-based approach (b) on the dependency parsing task. The structure in red is the output of the parser.

A general order prediction model is first trained on large scale unlabelled source language data, then for each structured prediction task, we distill the knowledge from the general model to teach task specific reordering modules.

We evaluate our method on three zero-shot cross-lingual structured prediction tasks (dependency parsing, part-of-speech and morphological tagging). By taking English as source language, we observe no obvious monolingual result loss in most cases, while the cross-lingual results could be significantly improved. We also analyze reordering perplexities of different target languages and show their correlation with performances of cross-lingual representation and structured prediction tasks. The conclusion reflects that both word order and word form could be important in cross-lingual learning.

2 Method Overview

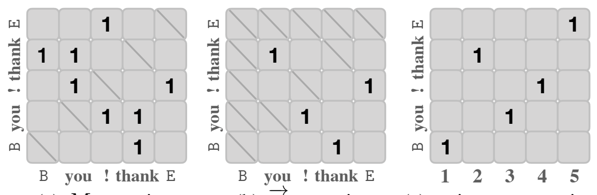
Given a sentence $x = \mathtt{B}, w_1, \cdots, w_n, \mathtt{E}$ (Begin and End are synthetic marks), we can have two views on x, a bag-of-words ω and an order of words o. Structured prediction tasks map x to a linguistic structure y (e.g., a parse tree). Here, we consider a *strict* zero-shot cross-lingual setting: given a labelled source language corpus $S = \{(x_s, y_s)\}$, we train a structured prediction model on S, and apply it directly on a target language sentence x_t (with ω_t, o_t), without seeing any labelled or unlabelled target language data. We will assume a reasonable cross-lingual word representation which maps source and target language words into a same vector space.

Usually, we use S to estimate probability $p(y|x_s) = p(y_s|\omega_s, o_s)$ which takes *word order as an input*. Models use word order o_s as a natural reference to design hyper-parameters (e.g., it suggests the linear chain structure of RNN and CRF) or extract order sensitive features (e.g., position embeddings in Transformer (Vaswani et al., 2017)). However, the target language may hold a different word order, so this tight connection between word order patterns and model architectures is inflexible for adaptation.

Here, we propose to estimate $p(y, o_s|\omega_s)$ instead of $p(y|\omega_s, o_s)$ (i.e., *moving the word order to the output*). We can conceptually factorize $p(y, o_s|\omega_s)$ to $p(y|\omega_s)p(o_s|\omega_s)$, where $p(y|\omega_s)$ is a structured prediction module, and $p(o_s|\omega_s)$ is a *reordering* module whose job is to recover a *source* language word order from bag-of-words ω (of both source and target language). Two modules share internal hidden states and model parameters. Comparing with $p(y|\omega_s, o_s)$, we observe that,

- $p(y|\omega_s)$ decouples word order and model architectures: we are now free to use order-insensitive models (e.g., Transformer without position embeddings). Instead of being in hyper-parameters, word order information now appears in the (shared) model parameters of $p(y|\omega_s)$ and $p(o_s|\omega_s)$.
- $p(o_s|\omega_s)$ plays the reordering action implicitly. When training on the source language, it guides the shared parameters to derive a correct order o_s from ω_s. When testing on the target language, since words in ω_t are in the same space of those in ω_s, the learned parameters in $p(o_s|\omega_s)$ will implicitly encode ω_t with a proper source language word order (like what it does for ω_s). As a consequence, $p(y|\omega_t)$ will receive a more familiar order information even if o_t is different from o_s.

Figure 1 shows an illustration of the two methods. Before moving to details of $p(y, o_s|\omega_s)$, we describe possible representations of word order.



(a) M_{ij} matrix (b) $\overrightarrow{M}_{ij}$ matrix (c) assignment matrix

Figure 2: Three order objectives of a sentence "B thank you ! E". A empty element is equal to 0. (a) An undirected adjacent matrix M. (b) A forward adjacent matrix $\overrightarrow{M}$. (c) An assignment matrix M' assigns the corresponding position of each word.

3 Word Order Representation

To share more parameters with downstream structured prediction tasks, we consider the word order recovery task as another (linear) structured prediction task. For two words w_i, w_j in the bag of words of a sentence x, we design two kinds of order objectives based on the adjacency matrix,

- undirected adjacency matrix M, where $M_{ij} = 1$ means that w_i and w_j are adjacent in x.
- directed adjacency matrices, $\overrightarrow{M}$ (forward) and $\overleftarrow{M}$ (backward), where $\overrightarrow{M}_{ij} = 1$ ($\overleftarrow{M}_{ij} = 1$) means that w_i is the previous (next) word of w_j,

Besides, as suggested by Mena et al. (2018), another word order objective is assignment matrix M'. It assigns each word to its correct position in a shuffled sentence. They show that assignment matrix has a strong ability to restore sequence from a random order. All of the above objectives are in the form of word-to-word matrices (Figure 2). They can easily supervise the same word-to-word self-attention matrices in Transformer Encoder.

4 Joint Prediction and Reordering

We now describe the implementation of $p(y, o_s|\omega_s)$. Basically, it requires the input to be a bag of words without order information. Here we choose a Transformer Encoder **without position embeddings** [2] (Vaswani et al., 2017) instead of order-sensitive networks like RNNs.

Input Features Denote $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ as each word's vector representation. We obtain $\boldsymbol{x}_i$ by concatenating a fixed cross-lingual word embedding $\boldsymbol{w}_i$ (from MUSE or mBERT), an optional cross-lingual part-of-speech tag embedding $\boldsymbol{t}_i$ (used only for dependency parsing task), and a trainable oc-

[2]It can be seen as an order-insensitive graph neural network.

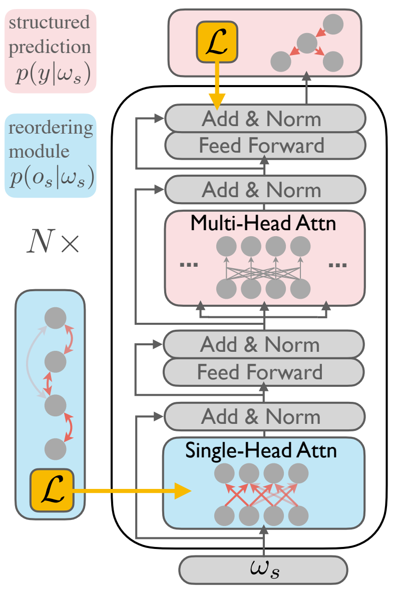


Figure 3: A diagram of a reordering-based Transformer encoder for dependency parsing task. A blue block is the reordering block. A red block is the structured prediction block. The yellow block is the supervision of reordering or structured prediction module.

currence index embedding $\boldsymbol{c}_i$. The $\boldsymbol{c}_i$ avoids the isomorphism problem [3] of *same words* by indicating which of them is used first, which is used second, and so on.

Transformer Encoder A Transformer Encoder is obtained by stacking N layers of identical Transformer blocks. We follow the standard notation to introduce how a Transformer block outputs a deep representation $X' = [\boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_n]$ based on a shallow input $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$. We start by mapping the input X to three spaces via linear transformations:

$$Q = W_Q \cdot X \quad K = W_K \cdot X \quad V = W_V \cdot X$$

$$A = \text{softmax}\left(\frac{Q \cdot K^\top}{\sqrt{d}}\right) \tag{1}$$

A self-attention matrix A is then obtained by scaled dot-product attention function. Where d is the column dimension of Q, A_{ij} measures an interaction score between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The self-attention output $O = A \cdot V$ contains an average of vectors X using weights from A, and the vector $\boldsymbol{o}_i$ in O is a deep hidden representation of $\boldsymbol{x}_i$. We can extend the above single-head self-attention layer to multi-head by defining multiple sets of $\{W_Q, W_K, W_V\}$. The output of a multi-head layer is the concate-

[3]Without $\boldsymbol{c}_i$, if there are two same words $\text{the}_1, \text{the}_2$ and any other word w_i in a sentence, the edge weight of $\langle w_i, \text{the}_1\rangle$ is always equal to $\langle w_i, \text{the}_2\rangle$.

nation of a set of O^k obtained by corresponding Q^k, K^k and V^k. We obtain the dimension-reduced output O through a linear transformation:

$$O = W_O \cdot \left(O^1 \oplus \cdots \oplus O^k\right)$$

$$X' = \texttt{Add\&Norm}(\texttt{FFN}(\texttt{Add\&Norm}(O)))$$

There is a residual connection and normalization (Add&Norm) layer after the self-attention layer, followed by a feed-forward layer, and finally another residual connection and normalization layer. After these layers we obtain a deep representation $X' = [\boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_n]$. The omitted detail of these three layers can be found in Vaswani et al. (2017).

Reordering Block A reordering block just adds a reordering signal $\mathcal{L}_{order}$ to the ordinary Transformer block. As mentioned in Section 3, $\mathcal{L}_{order}$ can guide the attention matrix A (in Equation 1), which is also a word-to-word matrix, to display a linear chain structure. After aggregating the shallow representation by using the matrix A as weights, the deep output X' of the reordering block has the contextual information with respect to the word order, since A is forced to match the desined word order. We test different setting of $\mathcal{L}_{order}$ with different word order repersentation matrices.

$$\mathcal{L}_{order} = \sum_{i,j} \begin{cases} (A_{ij} - \frac{1}{2}M_{ij})^2 \\ (A_{ij} - \frac{1}{2}(\overrightarrow{M}_{ij} + \overleftarrow{M}_{ji}))^2 \\ (A_{ij} - M'_{ij})^2. \end{cases} \quad (2)$$

In our experiments, we will select one of the reordering signals to train the model and compare it with each other. In addition, we use **single-head self-attention** to reduce computation because preliminary experiments show that multiple heads are not helpful for reordering blocks. Michel et al. (2019) has also shown that replacing multi-head with single-head does not hurt performance.

We cross stack reordering blocks with original Transformer blocks to build the complete encoder (Figure 3). The reordering blocks estimate probability $p(o_s|\omega_s)$ by learning linear word order on the source language. The original blocks estimate probability $p(y|\omega_s)$ by learning a structured prediction task without a direct reordering supervision. Given the bag of words ω_t in target language, thanks to cross-lingual word representation techniques, the reordering block predicts a similar word order $p(o_s|\omega_t)$ as source language. Since the word order of target language is not introduced, the encoder does not have word order drift between target language and source language.

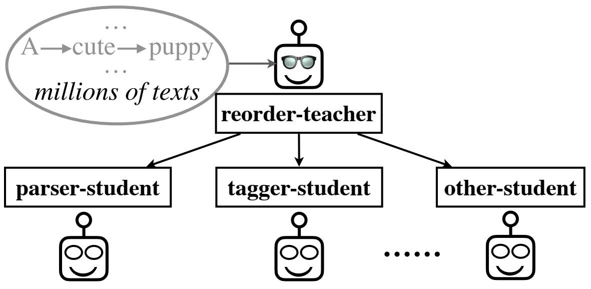


Figure 4: A diagram of teacher-students framework.

Downstream Classifier For a downstream task, we add a structured prediction classifier over the last encoder block. We investigate three downstream tasks. For graph-based dependency parsing (*dep*), we follow Dozat and Manning (2017) to use two bi-affine classifiers. For universal part-of-speech (*upos*) and morphological (*mor*) tagging, we use a multi-layer perceptron (MLP) classifier.

We train one of the downstream tasks by the corresponding cross-entropy loss function $\mathcal{L}_{down} \in \{\mathcal{L}_{dep}, \mathcal{L}_{upos}, \mathcal{L}_{mor}\}$. We assume the set of reordering block IDs is L. The joint objective function $\mathcal{L}_{joint}$ is to minimize a weighted combination:

$$\mathcal{L}_{joint} = \lambda_1 \mathcal{L}_{down} + \lambda_2 \sum_{l \in L} \mathcal{L}^l_{order}. \quad (3)$$

5 Distillation

In Equation 2, supervision signals on word reordering blocks are directly obtained from order objective (e.g., $\overrightarrow{M}$ and $\overleftarrow{M}$). Since we jointly perform reordering and structured prediction. The data for learning word order is constrained by the corpus size of structured prediction task (e.g, treebanks). On the other hand, there are massive unlabelled sentences which can help build a more powerful reordering module. To use those unlabelled data, one challenge is that directly feeding them into the joint learning model could be problematic since the severe imbalance between structured prediction signals and reordering signals would make the model focus on reordering. Furthermore, if we solve a different structured prediction task, we need to repeat the learning on those unlabelled data, which could be unnecessary and time-consuming. Therefore, it is important to separate the learning on those unsupervised data and sharing them efficiently.

Here we design a generic $p(o_s|\omega_s)$ model as a teacher and then distill the knowledge it learned in a large-scale unlabelled corpus into reordering blocks in a structured prediction task, which play the role of a student (Figure 4). The reordering student model attempts to learn the soft probability distribution from the teacher model, rather than the hard ground truth. Furthermore, multiple student models can share one teacher model, which avoids repeated training on a large-scale unlabelled corpus.

5.1 The Teacher Model

The **input features** of teacher model are the same as described in Section 4. We slightly modify the **Transformer encoder** by removing the feed forward layer to reduce GPU memory usage and speed up training on a large-scale corpus. There are only reordering supervision signals here, and all blocks of the encoder are reordering blocks.

Reordering Classifier We use a reordering classifier over the last encoder block's output ($X' = [\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_n]$) instead of the downstream classifier. To save space, we focus on describing the training and teaching of undirected adjacency matrix M order objective. [4].

We feed the vector $\boldsymbol{x}'_i$ into an MLP to obtain the dimension-reduced vector $\boldsymbol{v}_i$, and then compute the probability P_{ij} of word w_i and w_j being adjacent to each other by a bi-affine and sigmoid (σ) function.

$$\boldsymbol{v}_i = \text{MLP}(\boldsymbol{x}'_i)$$
$$P_{ij} = \sigma\left(\boldsymbol{v}_i^\intercal W \boldsymbol{v}_j + \boldsymbol{b}_1^\intercal \boldsymbol{h}_i + \boldsymbol{b}_2^\intercal \boldsymbol{d}_j \right), \quad (4)$$

where $\{W, \boldsymbol{b}_1, \boldsymbol{b}_2\}$ are parameters of the classifier.

Training and Teaching In training, the reordering blocks are supervised by the order objective like Equation 2, while the reordering classifier is supervised by cross-entropy loss function:

$$\mathcal{L}_{order}^{teacher} = -\sum_{i,j} M_{ij} \cdot \log(P_{ij}). \quad (5)$$

In teaching, the objective of the student model's reordering blocks is the teacher model's output instead of the hard order matrices. We modify Equation 2 as follows:

$$\mathcal{L}_{order}^{student} = \sum_{i,j} (A_{ij} - \frac{1}{2} P_{ij})^2 \quad (6)$$

6 Experiments

We demonstrate the effectiveness of our approach on three structured prediction tasks in a strict zero-shot cross-lingual setting, dependency parsing (DEP), universal part-of-speech tagging (UPOS), and morphological tagging (MOR).

We train our general reordering teacher and three structured prediction models on the train set of Universal Dependencies (UD) English-EWT treebank (v2.2) (Nivre et al., 2018). We use the development set and test set of the UD English-EWT treebank to validate source language performance. Following Ahmad et al. (2019a)'s setup, we take 30 other languages as target languages, and use the development set and test set of their treebanks to evaluate target languages performance.

For the reordering model, a `Base` train set is UD English, and an `Extra` set is automatically annotated raw texts (Ginter et al., 2017) generated by UDPipe v2.0 (Straka and Straková, 2017) from CommonCrawl and Wikipedia. Each sentence is automatic tokenization and syntactic annotations (include UPOS).

The hyperparameters we used in word reordering task and downstream tasks are summarized in Appendix B. The statistics of the UD treebanks are summarized in Appendix C.

6.1 Performances of the Reordering Model

Our Models and Baselines We explore the input features' influence, order representations , and unlabeled data size to the reordering model. For input features, we utilize MUSE, mBERT, and optional *upos* features. For order representations, we utilize an undirected (M) or directed ($\overrightarrow{M}, \overleftarrow{M}$) adjacency matrix, and an assignment matrix (Mena (2018)). For unlabeled data size, we gradually increase the training set in 10% increments.

Evaluation We use exact matching (`EM`) to measure reordering at the sentence level, which means the whole sentence to be correctly decoding, use order perplexity (`PPL`$= 2^{H(x)}$) to measure the cross entropy $H(x)$ of sentence x [5]. The cross-entropy of a sentence usually decomposes to some local n-grams, so `PPL` can more finely measure partial matching. During testing, we exclude sentences that are longer than 20 (only for `PPL`).

Results Firstly, we explore different order representations including M, $\overrightarrow{M} + \overleftarrow{M}$, and Mena (2018) (Table 1). We first compare two adjacency matrix objects. $\overrightarrow{M} + \overleftarrow{M}$ is better in both `EM` (+12.6) and

[4]Descriptions of the other two order objectives are placed in the Appendix A.

[5]Here the form of $H(x)$ is the same as at training, e.g. $H(x) = -\frac{1}{n}\sum_i \log \vec{P}_{i\,(i+1)}$ of $\overrightarrow{M}$ matrix. The bound of the `PPL` is [1, 20].

Mena (2018) EM/PPL	M EM/PPL	$\overrightarrow{M}+\overleftarrow{M}$ EM/PPL
47.2/4.90	46.8/4.26	**59.4/3.65**

Table 1: Development set performances of three order objects trained in the `Base` training set. Input feature contains MUSE embeddings and *upos* tags.

Reorder EM/PPL	MUSE	MUSE +upos	mBERT	mBERT +upos
`Base`	48.2/4.01	59.4/3.65	45.8/3.78	56.8/3.78
+10%	55.6/3.64	64.5/2.64	53.9/3.61	64.1/2.87
+20%	60.7/3.14	67.7/2.21	58.3/3.06	65.8/2.52
+30%	60.8/2.75	68.2/2.08	58.6/2.93	66.1/2.55
+40%	**61.0**/2.52	68.5/2.04	59.6/2.83	**66.3/2.25**
+50%	**61.0/2.48**	**68.9/1.89**	60.2/**2.77**	**66.3**/2.37
+60%	60.8/2.61	68.8/1.93	**60.4**/2.98	65.9/2.35

Table 2: The development results of the four input features using different amounts of data. "+10%" means additional using 10% `Extra` data and so on.

`PPL` (-0.61) mainly because of the stronger scoring function. $\overrightarrow{M}+\overleftarrow{M}$ uses two bi-affine functions to calculate forward and backward scores respectively, while M uses only one dot product scoring function in order to satisfy symmetry. Previous work (Dozat and Manning, 2017, 2018) shows bi-affine function outperforms dot product, and using two scoring functions can benefit from ensemble learning. Next we compare adjacency with assignment matrix, $\overrightarrow{M}+\overleftarrow{M}$ also outperforms Mena (2018) (+12.2 `EM`, -1.24 `PPL`), since the assignment matrix doesn't represent meaningful edges of graph while it also uses only one scoring function like M.

Secondly, since different downstream tasks assume different input features, we list four cases containing two cross-lingual word embeddings and conditions on whether to use extra *upos* features (Table 2). Basically, using extra *upos* features can improve the results because it alleviates the problem of out-of-vocabulary and low frequency words. Surprisingly, the mBERT representation formally carries some order information from the positional encoding in itself, but the reordering results based on mBERT is lower than MUSE. We guess this may be due to the loss of some lexical and order information when doing the subword-to-word conversion.

Finally, since reordering is an unsupervised task, we analyze the impact of the number of `Extra` texts used (Table 2). Two phenomena are observed. One is the more unannotated data, the better reordering performance because of the improved expression capacity of neural networks. The other is it's more difficult to improve reordering after using up to 50% data. This is because the reordering task is difficult in some cases. For example, "I like apples, bananas and oranges", any exchange of three fruits is a reasonable sentence. In the future, reordering models could track the progress of graph neural networks and further improve performance.

6.2 Results of The Downstream Tasks

Our Baselines We have five benchmark models from previous work (Ahmad et al., 2019a):

- RNN uses biLSTM encoder,
- Abs uses Transformer encoder with absolute position embedding,
- Rel uses undirected relative position embedding,
- NoP drops position embedding directly.
- mBERT is a direct fine-tuning method based on the pre-trained language model mBERT.

Our Models We have two variants of our model:

- Reord represents pipeline the reordering model and structured prediction model. Directly feed the reordered sequence into Rel model.
- noDst models two tasks as a multi-task learning task without distillation, with use 10% `Extra` data.

Our final two models use MUSE and word-level mBERT (base version) as word representations, respectively. Note that our mBERT-based model stacks 4-layers modified Transformer encoder on the top of mBERT encoder.

Evaluation First, we have three downstream task performance metrics. For DEP, following Ahmad et al. (2019a), we evaluate it with label attachment score (`LAS`) with punctuations excluded [6]. For UPOS, we evaluate it with token-level accuracy (`Acc`). For MOR, we evaluate it with token-level exact match (`EM`).

Second, we discuss three language distance metrics. Smith et al. (2017) report the "precision@5" of MUSE in a bilingual dictionary, which measures the word form distance (S17). Ahmad et al. (2019a) use the Manhattan distance of directional feature vector [7], which measures the word order distance

[6]The tokens labeled "PUNCT" or "SYM" POS tags are not included.

[7]They statistic the relative frequency of modifier before head in triples "(ModifierPOS, HeadPOS, DependencyLabel)"

DEP	Dist. to en			MUSE							mBERT	
	A19	S17	PPL	RNN	Abs	Rel	NoP	Reord	noDst	Our	mBERT	Our
en	0.00	1.00	1.89	88.30	**89.10**†	88.45	54.27	88.45	86.27	88.16	**90.73**†	90.58
no	0.06	0.85	2.49	72.83	72.75	72.81	31.90	73.04	70.15	**73.65**†	71.72	**73.48**†
it	0.12	0.86	2.50	**76.23**†	75.37	75.82	30.22	75.52	73.48	75.74	**79.06**†	78.34
fr	0.09	0.86	2.59	**73.46**	72.23	72.78	32.53	73.26	71.58	73.43	72.04	**73.40**†
pt	0.09	0.86	2.65	67.98	67.44	67.75	26.34	**68.50**†	66.07	68.39	67.13	**67.21**
es	0.12	0.85	2.70	66.91	66.71	66.44	28.47	67.19	64.73	**67.23**	65.30	**65.44**†
da	0.10	0.84	2.77	68.81	67.62	67.87	22.37	68.54	66.98	**68.99**	**70.69**†	69.40
pl	0.13	0.79	2.86	58.59	59.69	**62.23**†	27.35	60.92	59.08	61.13	63.05	**63.11**
sv	0.07	0.82	2.89	73.49	72.71	73.17	36.91	73.68	71.40	**73.78**	70.15	**71.23**†
nl	0.14	0.83	2.91	60.11	60.29	60.26	20.04	60.16	59.82	**60.82**†	**65.62**†	63.62
ca	0.13	0.81	2.92	65.57	65.30	65.13	21.11	65.47	64.58	**66.16**†	**66.47**†	65.96
cs	0.14	0.81	2.98	52.80	52.25	**53.80**†	16.34	53.34	50.63	53.68	50.26	**51.09**†
ru	0.14	0.78	3.01	50.81	50.44	51.63	15.91	51.35	48.65	**51.64**	**52.90**†	52.35
fi	0.20	0.80	3.02	48.74	48.21	48.69	10.06	48.71	47.74	**49.38**†	49.49	**50.13**†
de	0.14	0.75	3.09	59.31	60.58	61.62	19.44	61.44	59.38	**61.92**†	**63.51**	63.50
uk	0.13	0.75	3.10	51.14	52.06	52.28	17.05	52.61	50.37	**52.98**†	**55.94**†	53.39
ro	0.15	0.81	3.15	52.11	52.07	**54.10**†	15.23	52.43	51.78	53.39	46.40	**51.05**†
id	0.17	0.81	3.33	42.09	42.25	43.52	16.94	**43.83**†	40.82	43.43	42.99	**43.58**†
hr	0.13	0.75	3.40	50.67	50.03	52.86	12.35	53.53	48.09	**53.56**	56.08	**56.96**†
sl	0.13	0.77	3.52	54.57	54.70	56.54	13.32	56.65	51.92	**56.86**†	56.82	**56.95**†
sk	0.17	0.75	3.63	56.98	56.87	**58.15**	18.18	58.07	54.64	58.12	50.78	**55.71**†
bg	0.14	0.77	3.65	66.68	66.56	68.21	21.98	69.02	62.98	**69.22**†	71.24	**71.64**†
he	0.14	0.45	3.99	46.93	46.97	48.00	14.13	48.83	45.78	**48.95**	48.02	**48.26**†
et	0.20	0.73	4.03	44.40	43.93	44.87	13.62	45.06	40.71	**45.42**	46.40	**46.91**†
lv	0.18	0.68	4.11	49.59	48.86	49.30	16.98	49.13	45.95	**49.62**	45.10	**45.87**†
ar	0.26	0.69	4.44	25.48	25.64	28.04	8.17	29.26	23.58	**29.78**†	**30.34**	30.26
zh	0.23	0.68	4.56	10.77	10.59	11.36	7.10	12.14	10.69	**12.60**†	15.4	**17.18**†
hi	0.40	0.58	4.84	21.41	22.98	26.52	9.07	28.39	20.82	**28.74**†	15.48	**19.25**†
ko	0.33	0.58	5.07	15.40	16.06	16.40	6.63	17.68	13.46	**18.06**†	**19.10**†	18.22
la	0.28	0.34	5.11	33.91	33.42	35.21	12.89	35.32	29.83	**35.75**†	29.21	**30.39**†
ja	0.49	0.44	6.04	10.06	9.86	10.53	7.45	11.92	10.19	**11.97**	13.42	**16.34**†
Avg.	0.17	0.74	3.51	50.93	50.81	51.86	18.34	52.17	49.20	**52.48**	51.67	**52.34**

Table 3: LAS scores of our models and baseline models on 31 test sets of DEP task. '†'means that the best transfer model is statistically significantly better (by paired bootstrap test, $p < 0.05$) than others.

(A19). We use the PPL of target languages in the reordering model as distance, which measures the confidence of reordering model to restore target order. All distances are calculated by source (en) to 30 targets.

Results Firstly, we compare our model with some benchmark models on DEP task (Table 3). The first part is MUSE-based models, and overall, our hard reordering approach achieves competitive performance on the source language and soft reordering approach achieves the best cross-lingual performance on 21 languages. First, the Rel model achieves the highest cross-lingual performance in 4 baselines because of weakening the order by using a undirected relative position. The results of the benchmark model demonstrate that word order is indeed a trade-off of source and cross-lingual performance. Second, we analyze the effectiveness of multi-task learning. Comparing the original Rel model with the Reord model, we observe an improvement in cross-lingual performances, and this approach does not affect English performance at all. This illustrates the effectiveness of the reordering model, increasing the similarity between target sentences and source sentences. Nevertheless, the Reord model is still weaker than our approach. The main reason is that combining the reordering model in a pipline way can cause error propagation problems which affect cross-lingual performance. Third, we analyze the effectiveness of the distillation method. Comparing noDst with Abs, we observe that noDst performs worse in both English and cross-lingual results. This result shows that multi-tasking learning is affected by the data imbalance problem of the reordering task and DEP tasks. The model overfits the reordering task which has large amounts of data. In fact, it makes the parsing performances even weaker than single-task learning setting. It demonstrates the effectiveness of the proposed knowledge transfer approach.

as the directional feature.

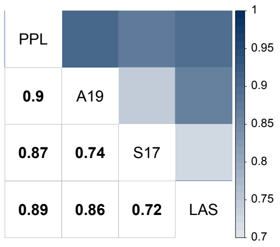


Figure 5: Pearson correlation coefficient for any two metrics of A19, S17, PPL and LAS. LAS is the cross-lingual performance of RNN model in DEP test set. We pick RNN because it's widely used and implicitly relies on word order.

Task		RNN	Abs	Rel	NoP	Our	mBERT	Our
UPOS	♡	**94.2**	93.9	93.3	56.2	93.7	**95.2**	**95.2**
	♠	37.8	41.3	44.6	12.6	**47.7**	74.5	**75.4**
MOR	♡	94.6	**94.8**	94.1	71.7	94.4	**95.7**	95.3
	♠	38.1	38.3	40.0	25.6	**41.1**	68.5	**69.0**

Table 4: `Acc`/`EM` scores of our models and baseline models on UPOS/MOR test sets. The ♡ indicates source language while ♠ indicates the average of 30 target languages to save space.

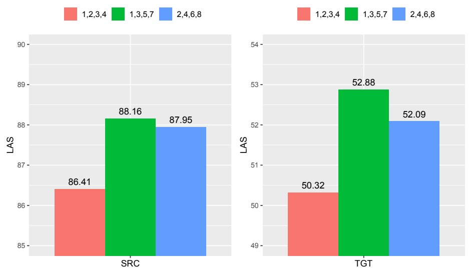


Figure 6: The impact of three different ways of combining *reordering* modules. The numbers in the legend indicate the sub-layer indices where the reordering module is located.

The second part is encoding with mBERT representation. Overall, our approach outperforms the mBERT encoder, This shows our method also works for mBERT representation. We analyze two cross-lingual representations by comparing the mBERT baseline with MUSE-based baselines (RNN, Abs, Rel), it shows that the zero-shot transfer performance with mBERT embeddings is weaker than MUSE on the DEP task, and that may because the mBERT representation is not well aligned across languages. Previous work (Wang et al., 2019) also demonstrates this conclusion, and they use parallel corpus of source and target language pairs as supervision to learn better alignment of contextual cross-lingual representations. Since using parallel corpus is not strictly a zero-shot setting, we do not compare this work, but our model is compatible with their approach and can benefit from the better aligned representations.

Secondly, we discuss the correlation among language distance metrics (A19, S17, PPL), and their correlation to the parsing performance (LAS). (Figure 5). Basically, all three distances are positively correlated with LAS. We observe a clearly higher correlation between A19 and LAS than S17 and LAS, possibly because the impact of word form to LAS is weaker than word order. We observe a slightly higher correlation between PPL and LAS than A19 and LAS, which shows that, the reordering module does learn word order information. We also observe a clearly higher correlation between PPL and S17 than A19 and S17, since the reordering model uses MUSE as input and order as object, it can specifically co-represent the distance of word form and word order, which may indicate that it's not enough to consider word order alone and the word form will helps.

Thirdly, we report results on the UPOS task and the MOR task (Table 4). Overall, our approach further improves cross-lingual results on both tasks over strong MUSE (+3.1 `Acc`/+1.1 `EM`) or mBERT (+0.9 `Acc`/+0.5 `EM`) baselines. This suggests that the reordering module is generally effective for different tasks and different cross-lingual word embeddings. In particular, RNN model achieves the best source language performance on UPOS task, which may indicate that RNN is able to capture better word order information than absolute position embedding. Comparing the MUSE-based cross-lingual results in the DEP task, UPOS and MOR have a worse performance, which suggests that *upos*, as a well-aligned cross-lingual feature, is useful for zero-shot transfer. However, the improvement after using the mBERT embeddings is more significant (e.g. from 47.7 to 74.5 `Acc`) than DEP task, the reason may be that UPOS and MOR task relies more on local information.

Finally, we analyze the reordering module by exploring three versions of the odd, even, and bottom layers (Figure 6). We observe that odd is the most reasonable version, as it incorporates order information evenly in each layer's representation. The even decreases performance less because it is only an offset of the odd layer, and we think that the main part of the decrease is due to the reordering supervision in the last layer, which slightly hurts the learning process of structured prediction. The bottom approach has the worst performance, and we guess the reason is that the high level representations are unable to receive fresh word order information.

7 Related Work

There has been a lot of recent research on the cross-lingual transfer of structured prediction tasks, including dependency parsing (Wang and Eisner, 2018a; Ahmad et al., 2019b), and POS tagging (He et al., 2019; Kim et al., 2017). Early work built delexicalized models to direct transfer, but at the expense of performance (McDonald et al., 2011; Rosa and Žabokrtský, 2015). With the development of cross-lingual word embedding techniques (Conneau et al., 2017; Devlin et al., 2019), the recent work has utilized it to retain lexical information (Ahmad et al., 2019b; Wang et al., 2019).

Data augmentation can enrich the word order that appears on the source language, thereby increasing the intersection with the target language's word order. Tiedemann and Agic (2016); Wang and Eisner (2016, 2018a) create a high-quality synthetic treebank to increase source data. But data augmentation requires expert knowledge to build treebank and extra train time. It does not apply to a larger number of target languages. Annotation projection relies on cross-language annotation mapping using parallel corpus and automatic alignment (Rasooli and Collins, 2015; Agić et al., 2016; Plank and Agić, 2018). Our approach does not require the above resources, only the source language's raw data.

Tiedemann et al. (2014); Wang and Eisner (2018b); Zhang et al. (2019); Rasooli and Collins (2019) reorder the source treebanks to make them similar to the target language of interest before training on the source treebanks. This is a source-to-target reordering that requires the use of parallel corpus or automatic alignment tools. Instead, we are target-to-source reordering, and training only at source language.

Some previous work has seen word order as a trade-off, Ahmad et al. (2019a) modified transformer encoder by using undirected relative position to learn weak order information. Liu and Fung (2020) using Conv1d to capture local word order and taking the positional embeddings from mBert to initialize a frozen positional embeddings. Our reordering module can fully learn the source language word order based on the bag-of-words input. It's useful for tasks that are sensitive to global word order, such as parsing.

8 Conclusion

This work focus on the source-to-target word order adaption in zero-shot transfer. We build structured prediction models containing a novel reordering module with a bag of words input. The reordering module is distilled from a task-generic, unsupervised, and large-scale pre-trained reordering teacher. Experiments show that, our model can significantly improve cross-lingual performances on three tasks without obviously hurting source language performance. Future work contains two parts: extending to multi-source transfer and extending to more structured prediction tasks such as NER which requires span-level reordering.

Acknowledgments

The authors wish to thank the reviewers for their helpful comments and suggestions, thank Peng Li and Yi Gao for their comments on writing. This research is funded by the NSFC (62076097) and the 2020 East China Normal University Future Scientists and Outstanding Scholars Incubation Programme (WLKXJ2020). The corresponding authors are Tao Ji, Yuanbin Wu and Xiaoling Wang.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019a. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. 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 2440–2452, Minneapolis, Minnesota. Association for Computational Linguistics.

Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Kai-Wei Chang, and Nanyun Peng. 2019b. Cross-lingual dependency parsing with unlabeled auxiliary languages. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 372–382, Hong Kong, China. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing.

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.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Junxian He, Zhisong Zhang, Taylor Berg-Kirkpatrick, and Graham Neubig. 2019. Cross-lingual syntactic transfer through unsupervised adaptation of invertible projections. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3211–3223, Florence, Italy. Association for Computational Linguistics.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838, Copenhagen, Denmark. Association for Computational Linguistics.

Zihan Liu and Pascale Fung. 2020. Do we need word order information for cross-lingual sequence labeling. *arXiv preprint arXiv:2001.11164*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 14014–14024.

Joakim Nivre, Mitchell Abrams, and et al. Agić. 2018. Universal dependencies 2.2. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Barbara Plank and Željko Agić. 2018. Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 614–620, Brussels, Belgium. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338, Lisbon, Portugal. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3845–3856. Association for Computational Linguistics.

Rudolf Rosa and Zdeněk Žabokrtský. 2015. KLcpos3 - a language similarity measure for delexicalized parser transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249, Beijing, China. Association for Computational Linguistics.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Jörg Tiedemann and Zeljko Agic. 2016. Synthetic treebanking for cross-lingual dependency parsing. *J. Artif. Intell. Res.*, 55:209–248.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140, Ann Arbor, Michigan. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Dingquan Wang and Jason Eisner. 2018a. Surface statistics of an unknown language indicate how to parse it. *Transactions of the Association for Computational Linguistics*, 6:667–685.

Dingquan Wang and Jason Eisner. 2018b. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337, Brussels, Belgium. Association for Computational Linguistics.

Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu, and Ting Liu. 2019. Cross-lingual BERT transformation for zero-shot dependency parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5721–5727, Hong Kong, China. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2019. Cross-lingual dependency parsing using code-mixed TreeBank. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 997–1006, Hong Kong, China. Association for Computational Linguistics.

Appendix for *Word Reordering for Zero-shot Cross-lingual Structured Prediction*

A Other Reordering Classifiers

For assignment matrix M' similar to M, we only modify the sigmoid function in Equation 4 to a column-normalised softmax function.

$$P_{ij} = \text{Softmax}_j\ (\ \boldsymbol{v}_i^\intercal W \boldsymbol{v}_j + \boldsymbol{b}_1^\intercal \boldsymbol{h}_i + \boldsymbol{b}_2^\intercal \boldsymbol{d}_j) \tag{7}$$

For directed adjacency matrices, we need to handle forward matrix $\overrightarrow{M}$ and backward matrix $\overleftarrow{M}$. Thus we use two MLPs to generate forward representation $\boldsymbol{h}_i$ and backward representation $\boldsymbol{d}_i$. With these two representations, the forward edge probability $\vec{P}_{ij}$ can be calculated by a bi-affine score function with a column-normalised softmax function, the backward edge probability $\overleftarrow{P}_{ij}$ can be calculated by a bi-affine score function with a row-normalised softmax function.

$$\begin{aligned} &\boldsymbol{h}_i = \overrightarrow{\text{MLP}}(\boldsymbol{x}'_i),\ \ \boldsymbol{d}_i = \overleftarrow{\text{MLP}}(\boldsymbol{x}'_i). \\ &\vec{P}_{ij} = \text{Softmax}_j\ \boldsymbol{h}_i^\intercal \vec{W} \boldsymbol{d}_j + \vec{\boldsymbol{b}}_1^\intercal \boldsymbol{h}_i + \vec{\boldsymbol{b}}_2^\intercal \boldsymbol{d}_j \\ &\overleftarrow{P}_{ij} = \text{Softmax}_i\ \boldsymbol{h}_i^\intercal \overleftarrow{W} \boldsymbol{d}_j + \overleftarrow{\boldsymbol{b}}_1^\intercal \boldsymbol{h}_i + \overleftarrow{\boldsymbol{b}}_2^\intercal \boldsymbol{d}_j \end{aligned} \tag{8}$$

Where $\{W, \vec{W}, \overleftarrow{W}, \boldsymbol{b}_1, \vec{\boldsymbol{b}}_1, \overleftarrow{\boldsymbol{b}}_1, \boldsymbol{b}_2, \vec{\boldsymbol{b}}_2, \overleftarrow{\boldsymbol{b}}_2\}$ appearing in Equations 7 and 8 are parameters.

B Hyper-parameters

The hyper-parameters we used in reordering teacher model (Table 5) and downstream structured prediction models (Table 6).

Layer	Hyper-parameter	Value
Input	MUSE	300
	mBERT	768
	POS	0.33
Transformer	Layer	6
	Hidden	512
	Head	1
	Dropout	0.2
MLP	$D_{in} \rightarrow D_{out}$	512→512
Trainer	Optimizer	Adam
	Learning rate	1e-3
	(β_1, β_2)	(0.9, 0.98)
	Batch size	80

Table 5: Hyper-parameters for reordering teacher.

C Details of Datasets

The statistics (number of sentences) of Universal Dependency (UD) treebanks are summarized in Table 7.

Layer	Hyper-parameter	Value
Transformer	Layer	8
	Hidden	200
	Head	8 or 1
	Dropout	0.2
Classifier	Bi-affine for arc	512→512
	Bi-affine for label	512→128
	MLP for POS&MOR	200→256
Multi-task	(λ_1, λ_2)	$(1.0, 0.25)$

Table 6: Hyper-parameters for downstream models.

Language	Name	#train	#dev	#test
Arabic	ar	6,075	909	680
Bulgarian	bg	8,907	1,115	1,116
Catalan	ca	13,123	1,709	1,846
Chinese	zh	3,997	500	500
Croatian	hr	6,983	849	1,057
Czech	cs	102,993	11,311	12,203
Danish	da	4,383	564	565
Dutch	nl	18,058	1,394	1,472
English	en	12,543	2,002	2,077
Estonian	et	20,827	2,633	2,737
Finnish	fi	12,217	1,364	1,555
French	fr	14,554	1,478	416
German	de	13,814	799	977
Hebrew	he	5,241	484	491
Hindi	hi	13,304	1,659	1,684
Indonesian	id	4,477	559	557
Italian	it	13,121	564	482
Japanese	ja	7,164	511	557
Korean	ko	27,410	3,016	3,276
Latin	la	15,906	1,234	1,260
Latvian	lv	5,424	1,051	1,228
Norwegian	no	29,870	4,300	3,450
Polish	pl	19,874	2,772	2,827
Portuguese	pt	17,993	1,770	1,681
Romanian	ro	8,043	752	729
Russian	ru	48,814	6,584	6,491
Slovak	sk	8,483	1,060	1,061
Slovenian	sl	8,556	734	1,898
Spanish	es	28,492	3,054	2,147
Swedish	sv	4,303	504	1,219
Ukrainian	uk	4,513	577	783

Table 7: Statistics of the UD dataset we used. We chose the same treebank as Ahmad et al. (2019a).