Improving Zero-Shot Dependency Parsing by Unsupervised Learning

Jiannan Mao[†‡*], **Chenchen Ding**[‡], **Hour Kaing**[‡],
Hideki Tanaka[‡], **Masao Utiyama**[‡], **Tadahiro Matsumoto**[†]
[†]Gifu University, Gifu, Japan
[‡]National Institute of Information and Communications Technology, Kyoto, Japan
[†]{mao, tad}@mat.info.gifu-u.ac.jp
[‡]{chenchen.ding, hour_kaing, hideki.tanaka, mutiyama}@nict.go.jp

Abstract

UDify (Kondratyuk and Straka, 2019) is a multilingual and multi-task parser fine-tuned on mBERT. It has demonstrated notable performance, even on few-shot languages. However, its performance saturates early and decreases gradually as training progresses with zero-shot languages. This work focuses on this phenomenon, which has not yet been sufficiently studied. Data augmentation methods based on unsupervised learning on monolingual data were designed to transform a zero-shot case into an artificial few-shot case. Experiments were conducted on the Breton language as a typical case study. For the Breton language, the unlabeled attachment score and other evaluation metrics were significantly improved. The parsing accuracies for other languages were not noticeably affected.

1 Introduction

A dependency parser can be efficiently trained on large treebanks when available (Dozat and Manning, 2017; Qi et al., 2020; Straka and Straková, 2020). For low-resource languages with limited treebanks, multilingual modeling has emerged as an efficient solution in which cross-lingual information is leveraged to compensate for the lack of data on specific languages. Ammar et al. (2016); Scholivet et al. (2019); Üstün et al. (2022) have demonstrated that the performance on part-of-speech (POS) or dependency parsing can be boosted by pairing languages with similarities. This multilingualism also cuts down the expense when training several models for a group of languages (Johnson et al., 2017; Aharoni et al., 2019).

UDify (Kondratyuk and Straka, 2019) is a multi-task self-attention network finetuned on multilingual BERT (mBERT) (Devlin et al., 2019) pre-trained embeddings, capable of producing annotations for any treebank from Universal Dependencies 2.3 (Zeman et al., 2018). UDify exhibits strong and consistent performance across a wide range of languages and tasks such as lemmatization, POS tagging, and dependency parsing. In addition to UDify, there are several comparable state-of-the-art methods, such as Choudhary (2021); Üstün et al. (2022); Langedijk et al. (2022); Effland and Collins (2023). A problem highlighted by several related studies is the substantial discrepancy in the performance of these methods in zero-shot learning scenarios, even with identical training strategies, datasets, models, and evaluation methods (Üstün et al., 2020; Langedijk et al., 2022).

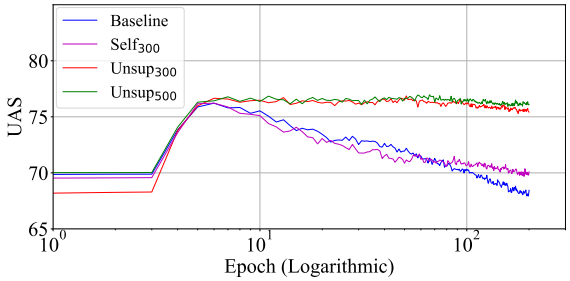


Figure 1: Change in the UAS of a model during the training process on the Breton test set for both the baselines (Baseline and Self) and the proposed method (Unsup).

This work investigates the underlying reason for the phenomenon regarding zero-shot cases by exhaustively examining the epochs during model training. To resolve this problem, a data augmentation strategy is proposed to improve the performance and stability of UDify. Specifically, the original UDify provides an initialization based on unsupervised learning for a zero-shot language; the generated results by unsupervised learning are then incorporated into UDify's training set so that the zero-shot language is converted into an artificial few-shot language.

Experiments on the Breton language are taken as a case study. The data augmentation, based

[*] This work was done during the first author's internship at National Institute of Information and Communications Technology, Kyoto, Japan.

on unsupervised learning, efficiently boosted the unlabeled attachment score (UAS) from 68.4% to 76.1%. Furthermore, the parsing accuracy for other languages did not decrease, which suggests that the overall robustness of multilingualism processing is still retained.

This paper is organized as follows. In Section 2, we first introduce UDify and unsupervised learning algorithms as the background of this study. Section 3 describes the performance of UDify in low-resource language scenarios, particularly in zero-shot languages, as well as the data augmentation method based on UDify and unsupervised learning algorithms. The experimental results of the case study, with discussions, are provided in Sections 4 and 5. Section 6 concludes the paper and discusses work scheduled for the near future.

2 Background

2.1 UDify

The UDify model jointly predicts lemmas, POS tags, morphological features, and dependency structures. The pre-trained mBERT model[1] is a self-attention network with 12 transformer encoder layers. It is used in the UDify model for cross-lingual learning. Specifically, mBERT's subword tokenization scheme is directly adopted in UDify without additional tags to distinguish the original languages. The tokens are then embedded and passed to the self-attention layers. A strategy similar to that of ELMo (Peters et al., 2018) is adopted, where a weighted sum of the outputs of all layers is computed as follows and fed to a task-specific classifier.

$$e_j^{task} = \eta \sum_i mBERT_{ij} \cdot softmax(w)_i$$

Here, e^{task} denotes the contextual output embeddings for the tasks. In the case of UDify, the tasks include lemmas, POS tags, morphological features, and dependency structures. The term $mBERT_{ij}$ represents the $mBERT$ representation for layer $i = 1, \cdots, 12$ at token position j. The terms w and η denote trainable scalars, where the former applies to mBERT and the latter scales the normalized averages. For words that have been tokenized into multiple subword units, only the first subword unit is fed to the dependency parsing classifier.

The dependency parsing classifier is a graph-based bi-affine attention classifier (Dozat and Manning, 2017). It projects the embeddings e_j^{task} through arc-head and arc-dep feedforward layers. The resulting outputs are combined using bi-affine attention to produce a probability distribution of the arc heads for each word. Finally, the dependency tree is decoded using the Chu–Liu/Edmonds algorithm (Chu, 1965; Edmonds et al., 1967).

[1] github.com/google-research/bert/multilingual.md

2.2 Unsupervised Dependency Learning

Adhering to the properties of dependency syntax (Robinson, 1970), a general unsupervised algorithm for projective N-gram dependency learning (Unsupervised-Dep) was described in Ding and Yamamoto (2013, 2014). This method constructs the best dependency tree with a dynamic programming method using a CYK style chart and is based on the complete-link and complete-sequence non-constituent concepts. However, considering the time complexity of this approach for arbitrary N-gram dependency learning, which may not be ideal for practical applications, we chose to focus in this study on the case of the bi-gram.

When considering the bi-gram, the directionality of a complete-link is set by the outermost dependency relation, with $(w_i \rightarrow w_j)$ indicating a rightward link and $(w_i \leftarrow w_j)$ indicating a leftward one. A basic complete-link is an adjacent word dependency link.

A complete-sequence represents a null or sequential set of adjacent complete-links with identical directionality. It begins as a null sequence of complete-links based on a single word, its smallest constituent. The direction of a complete-sequence matches that of its component complete-links.

The unsupervised learning firstly constructs the complete-links and complete-sequences for a substring, and then incrementally merges the complete-links into larger complete-sequences and complete-sequences into larger complete-links, recursively defined as follows.

$$Link_r(i,j) \equiv \{(w_i \rightarrow w_j), Seq_r(i,k), Seq_l(k+1,j)\}$$

$$Link_l(i,j) \equiv \{(w_i \leftarrow w_j), Seq_r(i,k), Seq_l(k+1,j)\}$$

$$Seq_r(i,j) \equiv \{Seq_r(i,k), Link_r(k,j)\}$$

$$Seq_l(i,j) \equiv \{Link_l(i,k), Seq_l(k,j)\}$$

In these equations, $Link$ and Seq denote a complete-link and complete-sequence respectively, whereas r and l indicate the direction (right or left). Furthermore, i and j respectively represent the starting and ending indices of a word sequence in the sentence.

This process continues until the $Link_r(1, n)$ with the maximum probability has been constructed, where n is the index of the last word in the sentence. These probabilities are calculated using the Inside–Outside algorithm (Lari and Young, 1990). Finally, the Viterbi algorithm (Forney, 1973) is employed to determine the tree construction in the calculated Inside portion with the maximum probability, thus generating the optimal structure.

3 Proposed Method

3.1 Motivation

In this work, we adopt the classification of low-resource languages such as Yang et al. (2022), dividing them into two types: few-shot languages, which only contain a small number of training data, and zero-shot languages, which do not contain any training data. This categorization enables a more systematic examination of UDify's performance, providing the insights that guide our research.

Although the number of training epochs in the original UDify was set to 80, the reported experimental results in our reproduction could only be reached after around 200 epochs.[2] Therefore, we monitored the changes in the test sets of few-shot and zero-shot languages during these 200 epochs, which we detail in the following.

Few-Shot Languages UDify performs well even on few-shot languages that were not included in the mBERT training set. For instance, the Upper Sorbian language, which was not included in mBERT's training data, is very low-resourced, with the `Upper_Sorbian-UFAL` treebank consisting of a training set of only 23 sentences and no development set. Nevertheless, UDify still manages to demonstrate impressive parsing accuracy in Upper Sorbian and other few-shot languages. During the course of training, the parsing accuracy on the test set increases steadily, as illustrated in Figure 2.

Zero-Shot Languages In the Universal Dependencies 2.3 used for UDify's training, treebanks of 13 languages have no training or development sets. Among these languages, three[3] are included in mBERT's training data, and their dependency structures are learned by UDify through transfer learning. Unlike high-resource and few-shot languages, a decrease in the accuracy of dependency

[2] lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3042

[3] i.e., Breton, Tagalog, and Yoruba

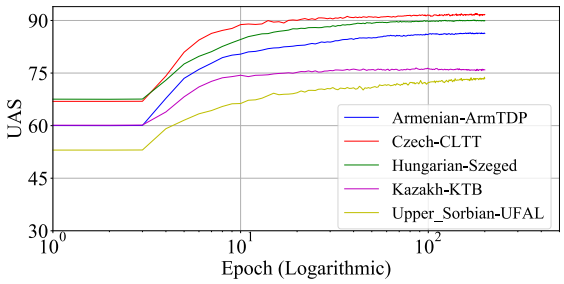


Figure 2: Changes in the UAS of few-shot languages during the training process.

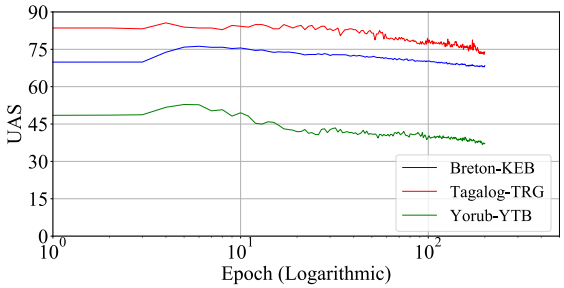


Figure 3: Changes in the UAS of zero-shot languages during the training process.

parsing is exhibited during the learning process of all of these zero-shot languages. The best performance typically was reached after 8 epochs of training, as shown in Figure 3.

This phenomenon has been noticed (Üstün et al., 2020; Choudhary, 2021; Langedijk et al., 2022; Üstün et al., 2022) but not yet systematically investigated to the best of our knowledge. Given that the accuracy of dependency parsing for zero-shot languages tends to decrease as training progresses, the number of epochs becomes a crucial factor in the inconsistency observed in related work.

Considering the observed positive correlation between parsing accuracy and the number of training epochs for high-resource and few-shot languages, it is somewhat unexpected to encounter a substantial discrepancy between the optimal and final testing results for zero-shot tasks. This calls for specialized strategies to bridge this gap. In the following text, we introduce a data augmentation technique based on Unsupervised-Dep. This technique aims to reduce the performance gap and thus improve the effectiveness of UDify in parsing dependency arc-heads for zero-shot languages.

3.2 UDify with Unsupervised Augmentation

When applying Unsupervised-Dep to data augmentation, it is essential to ensure the generated data

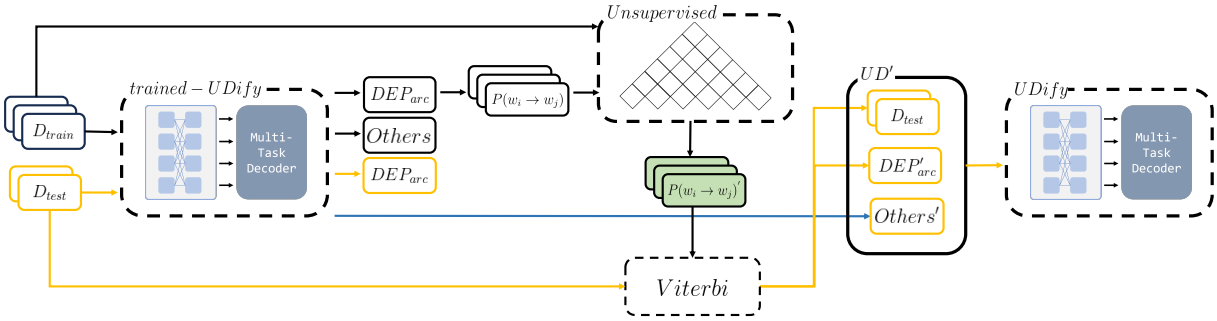


Figure 4: Integration of the parsing results from UDify with the training outcomes from the Unsupervised-Dep for new UDify model training. Models/algorithms are represented by dashed line boxes, whereas data/parameters are depicted within solid line boxes. Additionally, all processes associated with the construction of $UD^{'}$ are indicated by gold and blue lines.

aligns with the original data format. Therefore, in addition to dependency arc-heads, other types of data must be generated and combined with the results from Unsupervised-Dep. This ensures that the generated data conforms to the Universal Dependencies (UD) treebank format. The entire data augmentation process, as depicted in Figure 4, is described in the following paragraphs.

Unsupervised-Dep has a high time complexity of $O(n^3)$, which makes the common practice in the original methods, which start training from a random probability, somewhat inefficient. To circumvent this, we decided to leverage the parsing results from UDify to initialize the probabilities. Despite the potential decrease in UDify's accuracy on zero-shot languages during its training, the final results consistently outperform those from other parsing models (Qi et al., 2018; Zeman et al., 2018; Tran and Bisazza, 2019), providing a robust foundation for our initialization approach.

Our approach commences by feeding the raw corpus, denoted as D_{train}, into $trained\text{-}UDify$, generating the dependency arc-head, represented by DEP_{arc}. Next, statistical computations are performed specifically on the DEP_{arc} elements. The results, $P(w_i \rightarrow w_j)$ and $P(w_i \leftarrow w_j)$, serve as the initial probabilities for Unsupervised-Dep. Note that although the UDify parsing results do include other types of information such as POS and lemma (denoted as $others$), these are disregarded in the parsing results of D_{train} because they do not participate in our subsequent computations.

The input for Unsupervised-Dep consists of D_{train} and the initial probabilities $P(w_i \rightarrow w_j)$ and $P(w_i \leftarrow w_j)$. After undergoing a set number of epochs, we obtain the re-estimated probabilities $P(w_i \rightarrow w_j)^{'}$ and $P(w_i \leftarrow w_j)^{'}$. This process adheres to the principles of the Expectation–Maximization algorithm, as Algorithm 1 confirms. Notably, in the algorithm, the inside probability of $Link/Seq$ is denoted as α, whereas the outside probability of $Link/Seq$ is represented as β. These re-estimated probabilities serve as parameters for the inside part and are later integrated with the Viterbi algorithm, denoted as $Viterbi$. The implementation of $Viterbi$ allows the optimal dependency arc-head structure to be discovered.

Algorithm 1 Unsupervised-Dep

Input: $P(w_i \rightarrow w_j), P(w_i \leftarrow w_j)$,
$D_{train} = s_1, ..., s_n$
Output: $P(w_i \rightarrow w_j)^{'}, P(w_i \leftarrow w_j)^{'}$
for each $s \in D_{train}$ **do**
  for $i = 1$ to length(s) **do**
    for $j = 1$ to length(s) **do**
      $\alpha_r^{Link}(i,j), \alpha_l^{Link}(i,j) = inside$
      $(i, j, P(w_i \rightarrow w_j), P(w_i \leftarrow w_j))$
  for $i = 1$ to length(s) **do**
    for $j = 1$ to length(s) **do**
      $\beta_r^{Link}(i,j), \beta_l^{Link}(i,j) = outside$
      $(i, j, P(w_i \rightarrow w_j), P(w_i \leftarrow w_j))$
  for $i = 1$ to length(s) **do**
    for $j = 1$ to length(s) **do**
      $P(w_i \rightarrow w_j)^{'} = \frac{\alpha_r^{Link}(i,j)\beta_r^{Link}(i,j)}{P(Link_r(1,n))}$
      $P(w_i \leftarrow w_j)^{'} = \frac{\alpha_l^{Link}(i,j)\beta_l^{Link}(i,j)}{P(Link_r(1,n))}$

To construct the data in the UD treebank format, the parse results of D_{test} from $trained\text{-}UDify$ are required. However, we need to retain information other than DEP_{arc}, denoted as $others^{'}$. This information is used along with $DEP^{'}_{arc}$, which is generated by parsing D_{test} with the trained $Viterbi$ algorithm and searching for the optimal dependency

arc-head structure. At this point, all the elements required for constructing the new data in the UD format, denoted as $UD^{'}$, have been assembled. The constructed $UD^{'}$ is then combined with the existing UD treebank for the subsequent UDify training. Notably, the dependency arc-dep is only changed when the word is linked to changes to the root; in all other cases, it remains unchanged.

4 Experiments

4.1 Dataset

We selected Breton from OPUS (Lison and Tiedemann, 2016) as our target low-resource language for the implementation of Unsupervised-Dep. Breton, which is most often tokenized using spaces, is classified as a language facing serious risk of extinction (Parliament et al., 2014; Tyers and Ravishankar, 2018). After cleaning the collected data using the MOSES tool (Koehn et al., 2007), we obtained 99.5k sentences.

From this collection, we first identified a subset of 300 sentences, referred to as test_{300}. To conduct more detailed testing of the sentence structures generated by Unsupervised-Dep, we later expanded this subset by incorporating an additional 200 sentences, resulting in a total of 500 sentences, referred to as test_{500}. The division of the collected data in the experiment is summarized in Table 1. The training set is referred to as D_{train} in Section 3.2 and is used for obtaining and updating the probabilities $P(w_i \rightarrow w_j)$ and $P(w_i \leftarrow w_j)$. The validation set is applied to validate the results of Unsupervised-Dep learning. The test set, referred to as D_{test} in Section 3.2, demonstrates the results of our data augmentation. In subsequent sections, we refer to these collected data as OPUS-br.

To evaluate our proposed method, we used the same version of the UD treebank that UDify uses for our experiments. During training, we concatenated all training sets, mirroring the approach of McDonald et al. (2011). We shuffled all sentences before each epoch and fed the mixed batch of sentences into the network, including sentences from any language or treebank, whether they were original UD treebank sentences or those generated through Unsupervised-Dep.

4.2 Setup

To minimize the impact of potential experimental environmental variations (Popel and Bojar, 2018),

data set	#sentence	#word
train	90,000	1,023,292
valid	9,000	113,066
test_{300}	300	2,663
test_{500}	500	4,469

Table 1: Division of the OPUS-br into training, validation, and test sets, and the number of words in each set.

Hyper-parameters	Value
Dependency arc-dep dimension	256
Dependency arc-head dimension	768
Optimizer	AdamW
b1,b2	0.9,0.99
Dropout	0.5
Bert Dropout	0.2
Mask probability	0.2
Layer dropout	0.1
Batch size	32
Epochs	200
Base learning rate	$7e^{-3}$
mBERT learning rate	$5e^{-5}$
Learning rate warmup steps	8000
Gradient clipping	5.0

Table 2: Hyperparameter settings

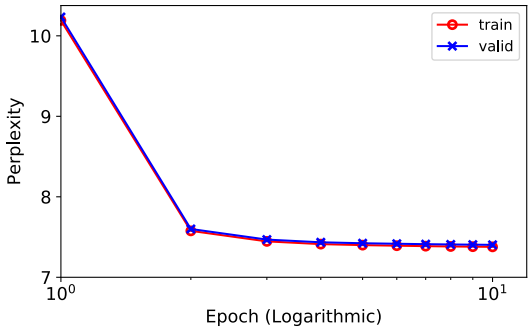


Figure 5: Training and validation set perplexity after each epoch.

we re-implemented the model provided by UDify.[4] To expedite the experiment, we implemented multi-GPU parallel training[5] by modifying UDify using Horovod (Sergeev and Balso, 2018). This modification also led to changes in the experimental parameters, which are detailed in Table 2. Furthermore, we divided the learning rate into two categories of base and mBERT, and set the learn-

[4] github.com/Hyperparticle/udify

[5] When parallelizing across seven V100s, replicating UDify requires close to 12 days.

	Breton					Other Languages				
	UPOS	UFeats	Lemma	UAS	LAS	UPOS	UFeat	Lemma	UAS	LAS
Baseline	68.6	47.8	44.7	68.4	48.5	83.6	83.2	91.7	77.7	70.6
Self_{300}	72.3	47.3	50.5	70.1	51.0	83.6	83.2	92.0	77.7	70.5
Unsup_{300}	74.3	**51.0**	57.6	75.4	60.5	83.6	83.2	92.0	77.7	70.6
Unsup_{500}	**74.7**	50.5	**57.7**	**76.1**	**61.5**	83.6	83.2	91.9	77.7	70.6

Table 3: Full UD scores on Breton and other languages obtained by different methods.

	UAS		gap
	last	best	
UDify	63.5	-	-
Baseline	68.4	76.2	-7.8
Self_{300}	70.1	76.2	-6.1
Unsup_{300}	75.4	76.5	-1.1
Unsup_{500}	**76.1**	**76.9**	**-0.8**

Table 4: UAS on Breton obtained using different methods. The UDify result was reported in Kondratyuk and Straka (2019).

ing rate for mBERT to $5e^{-5}$. This approach was proven effective by Kondratyuk and Straka (2019).

We initially computed the parsing results of the training data from OPUS-br using the re-implemented UDify model. These results served as the initial probabilities for $P(w_i \rightarrow w_j)$ and $P(w_i \leftarrow w_j)$. These probabilities were continuously re-estimated throughout the learning process of OPUS-br. To monitor the progression, we used the perplexity to observe the changes in $P(Link_r(1, n))$ using these re-estimated probabilities for both the training and validation set, as shown in Figure 5. After the 10th training epoch, we employed the re-estimated probabilities to parse the test set from OPUS-br.

In our multilingual parser experiments, we utilized a replicated UDify model, which we refer to as the Baseline. Our methods, Unsup_{300} and Unsup_{500}, integrate the refined dependency arc-heads generated by Unsupervised-Dep into the UD treebank. Unsup_{300} incorporates sentences from OPUS-br test_{300}, generated by Unsupervised-Dep, into the UDify training set. For Unsup_{500}, we used the sentences from OPUS-br test_{500}.

Inspired by the work of Rybak and Wróblewska (2018), we conducted an experiment with a comparative method referred to as Self_{300}. Within this approach, we used the test_{300} data, diverging from the use of Unsupervised-Dep for sentence refinement. The sentences were directly parsed with the replicated UDify model instead. The parsing results were then converted into UD treebank format, merged with the original training set, and used for a new round of UDify model training.

4.3 Result

A comparison with the experimental results reported in the original UDify study confirms that UDify was re-implemented successfully, as demonstrated in Table 4. The results in this table not only indicate a significant improvement in UDify's ability to avoid the decrease in dependency arc-head accuracy for the Breton language at the end of the training, regardless of the proposed method that was implemented but also indicate that while self_{300} causes only a minor increase in the UAS score, Unsup_{300} and Unsup_{500}, which incorporate data generated from Unsupervised-Dep, significantly augment the accuracy of the dependency arc-head. In addition, the changes in UAS for the Breton during the training process of UDify under different methods are shown in Figure 1.

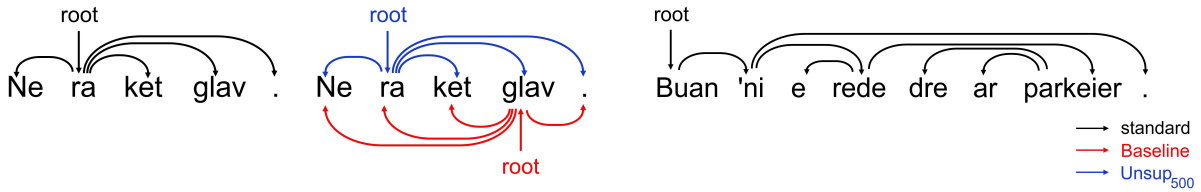
Given UDify's role as a multilingual and multi-task parser, it is vital to consider the influence of our proposed methods on other tasks and languages. For a comprehensive comparison, the remaining UD scores of Breton and other languages have been compiled in Table 3.

Considering all results, we argue that the conversion of zero-shot language learning into a few-shot language learning scenario is both essential and effective in multilingual modeling contexts.

5 Discussion

5.1 Dependency on the Zero-Shot Task

As shown in Table 4, transforming zero-shot languages into few-shot languages demonstrated a significant decrease in the performance disparity between the peak and final performance during the training of UDify. It is evident that our proposed method of data augmentation, Unsupervised-Dep,



a)-1. Standard

a)-2. Generation results of baseline and proposed

b). Example of sentence-ending punctuation with non-root parent

Figure 6: Examples of dependency structures in the Breton language.

can effectively mitigate the decline in accuracy during the training process.

Initially, considering that our data were manually created from raw data collected from OPUS, and because we had concerns about the quality of such data and constraints related to training time, we used test_{300}. After observing a positive impact on the experimental results from the sentences generated by Unsupervised-Dep, we expanded our experiment to use test_{500}. Because of time constraints, our current study did not definitively establish a correlation between the volume of data generated by Unsupervised-Dep and improved performance on zero-shot languages. We anticipate that future research will explore this potential relationship.

We now turn our attention to an analysis of specific instances within the parsing results. In a dependency structure, the terminus of an arc is referred to as the parent node, while the origin is deemed the child node. As illustrated in Figure 6a)-1, the parent node of "." is designated as "ra." In the majority of cases, the parent node of a sentence's terminal punctuation is the root of the sentence itself, as demonstrated in Figure 6a)-1. However, exceptions do exist, as exemplified in Figure 6b). An examination of the `Breton-KEB` test set revealed that such exceptions occurred 11 times under the standard method, 60 times under the Baseline, and 34 times under Unsup_{500}. When compared with those of the Baseline, the reduced number of exceptions in the Unsup_{500} test set signifies that our proposed method mitigates these irregularities.

In the parsing results, both the Baseline and Unsup_{500} correctly classified the UPOS of the keywords ("ra" as `VERB`, "glav" as `NOUN`). Here, "ra" translates to "do" and "glav" translates to "rain" in English, signifying "Do not rain" in Breton. However, the Baseline made an error in selecting the root, which was completely resolved in the Unsup_{500} results, as depicted in Figure 6a)-2.

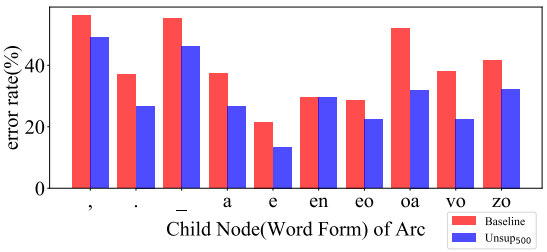


Figure 7: Percentage of child nodes connected to erroneous parent nodes under different methods, with the x-axis sorted by the count of child node errors in the Baseline.

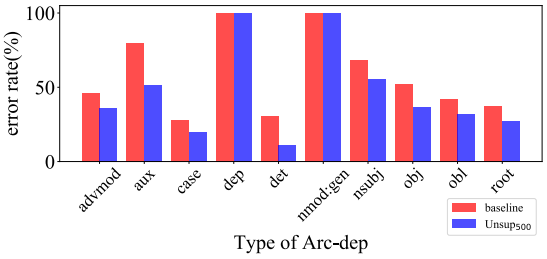


Figure 8: Percentage of arc-dep misclassifications under different methods, with the x-axis sorted by the count of arc-dep errors in the Baseline.

The status of child node connections also shows significant improvement. We display the proportion of child nodes incorrectly linked to parent nodes by both the Baseline and Unsup_{500} in Figure 7. Evidently, Unsup_{500} has made significant strides in the parsing outcomes of dependency arcs when compared with the Baseline. The labeled attachment score (LAS) is computed based on the UAS, and for Unsup_{500}, the LAS is 13 points higher than it is for the Baseline. Thus, we also display the proportion of incorrectly labeled arc-dep in the Baseline and Unsup_{500} in Figure 8.

As shown in Figure 8, two types of arc-deps, dep and `nmod:gen`, have an error rate of 100%, albeit

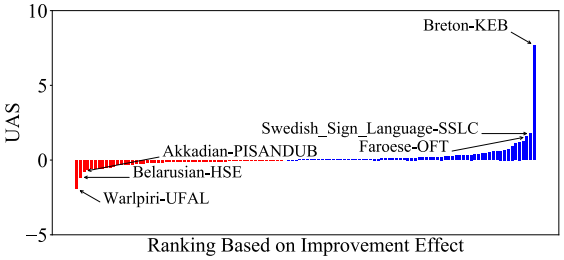


Figure 9: Changes in the UAS changes for the Baseline and Unsup_{500} models on all test treebanks. The x-axis sorts the UD treebanks by the ascending improvement of the proposed method over the Baseline.

Figure 10: Changes in the LAS for the Baseline and Unsup_{500} models on all test treebanks. The x-axis sorts the UD treebanks by the ascending improvement of the proposed method over the Baseline.

for different reasons. For dep, which only accounts for 0.3% of all arc-deps in all UD treebanks, it can be regarded as a tagging error.[6] By contrast, nmod:gen is an arc-dep type that only exists in Breton.[7] As our model is trained using existing data from UDify and raw Breton corpora collected from OPUS, it is incapable of correctly identifying the arc-dep type as nmod:gen.

5.2 Tasks on Other Languages

Considering UDify's role as a multilingual and multi-task parser, it is necessary to evaluate the impact of the proposed method on other tasks and languages. To observe in detail the differences and changes in the UAS and LAS between the Baseline and Unsup_{500}, we conducted tests across all treebanks and display the results in Figures 9 and 10. From these figures, it is evident that while Unsup_{500} has improved the UAS and LAS accuracy for Breton, it has had virtually no impact on the parsing precision of dependency constructions in other languages.

While our primary research objective was to improve the parsing accuracy of dependency arc-heads, we have also enhanced UDify's performance in other tasks for Breton, as evidenced by the data in Table 3. Given that UDify must balance the loss produced by multiple decoders during training and the work of Rybak and Wróblewska (2018), these variations in evaluation metrics are considered reasonable. From a broader perspective, our method has not had any adverse impact on other languages and tasks, maintaining their performance levels.

[6] universaldependencies.org/u/dep/dep.html
[7] universaldependencies.org/br/dep/nmod-gen.html

6 Conclusion and Future Work

This study investigated the issue of decreased parsing accuracy exhibited by UDify in zero-shot language scenarios, despite its generally outstanding performance in few-shot language scenarios. To address this problem, we proposed a method that applies an unsupervised algorithm to transition a zero-shot language into a few-shot language context, thereby effectively expanding the dataset and enhancing the model's learning capability.

The efficacy of our approach has been substantiated through our experimental results. By incorporating sentence dependency arc-head structures produced by Unsupervised-Dep into UDify's training data, we achieved a substantial improvement in UDify's performance with zero-shot languages. This improvement was significant even when only a limited number of sentences, such as 300 or 500, were used. Although our constraints did not allow for a definitive demonstration of a positive correlation between the number of sentences generated by Unsupervised-Dep incorporated into the training data and the improvement of UDify in zero-shot languages, this correlation remains a possibility.

In terms of future endeavors, our research objectives include: 1) investigating further the potential positive correlation between the volume of unsupervised generated data included in the training set and the subsequent improvement in UDify's performance on zero-shot languages and 2) exploring other contributing factors and considerations that may further enhance the performance of UDify in zero-shot language scenarios.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. 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 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Chinmay Choudhary. 2021. Improving the performance of UDify with linguistic typology knowledge. In *Proceedings of the Third Workshop on Computational Typology and Multilingual NLP*, pages 38–60, Online. Association for Computational Linguistics.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

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.

Chenchen Ding and Mikio Yamamoto. 2013. An unsupervised parameter estimation algorithm for a generative dependency n-gram language model. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 516–524, Nagoya, Japan. Asian Federation of Natural Language Processing.

Chenchen Ding and Mikio Yamamoto. 2014. A generative dependency n-gram language model: Unsupervised parameter estimation and application. *Information and Media Technologies*, 9(4):857–885.

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

Jack Edmonds et al. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Thomas Effland and Michael Collins. 2023. Improving low-resource cross-lingual parsing with expected statistic regularization. *Transactions of the Association for Computational Linguistics*, 11:122–138.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. 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 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Anna Langedijk, Verna Dankers, Phillip Lippe, Sander Bos, Bryan Cardenas Guevara, Helen Yannakoudakis, and Ekaterina Shutova. 2022. Meta-learning for fast cross-lingual adaptation in dependency parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8503–8520, Dublin, Ireland. Association for Computational Linguistics.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

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.

European Parliament, Directorate-General for Internal Policies of the Union, and M Prys Jones. 2014. *Endangered languages and linguistic diversity in the European Union*. Publications Office.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke

Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Martin Popel and Ondrej Bojar. 2018. Training tips for the transformer model. *Prague Bull. Math. Linguistics*, 110:43–70.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Jane J. Robinson. 1970. Dependency structures and transformational rules. *Language*, 46(2):259–285.

Piotr Rybak and Alina Wróblewska. 2018. Semi-supervised neural system for tagging, parsing and lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 45–54, Brussels, Belgium. Association for Computational Linguistics.

Manon Scholivet, Franck Dary, Alexis Nasr, Benoit Favre, and Carlos Ramisch. 2019. Typological features for multilingual delexicalised 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 3919–3930, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexander Sergeev and Mike Del Balso. 2018. Horovod: Fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.

Milan Straka and Jana Straková. 2020. UDPipe at EvaLatin 2020: Contextualized embeddings and treebank embeddings. In *Proceedings of LT4HALA 2020 - 1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 124–129, Marseille, France. European Language Resources Association (ELRA).

Ke Tran and Arianna Bisazza. 2019. Zero-shot dependency parsing with pre-trained multilingual sentence representations. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 281–288, Hong Kong, China. Association for Computational Linguistics.

Francis M Tyers and Vinit Ravishankar. 2018. A prototype dependency treebank for Breton. In *Actes de la Conférence TALN. Volume 1 - Articles longs, articles courts de TALN*, pages 197–204, Rennes, France. ATALA.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2022. UDapter: Typology-based language adapters for multilingual dependency parsing and sequence labeling. *Computational Linguistics*, 48(3):555–592.

Zhe Yang, Qingkai Fang, and Yang Feng. 2022. Low-resource neural machine translation with cross-modal alignment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10134–10146, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.