General Paper Peer-Reviewed

Data Augmentation for Low-Resource Languages in Multilingual Dependency Parsing

Jiannan Mao[a], Chenchen Ding[b], Hour Kaing[b], Hideki Tanaka[b], Masao Utiyama[b] and Tadahiro Matsumoto[a]

UDify (Kondratyuk and Straka 2019) is a multilingual, multi-task parser fine-tuned on mBERT that achieves remarkable performance on high-resource languages. However, on some low-resource languages, its performance saturates early and decreases gradually as training proceeds. To address this issue, this study applies a data augmentation method to improve parsing performance. We conducted experiments on five few-shot and three zero-shot languages to test the effectiveness of this approach. The unlabeled attachment scores were improved on the zero-shot language dependency parsing tasks, with the average score increasing from 55.6% to 59.0%. Meanwhile, dependency parsing tasks in high-resource languages and other Universal Dependencies tasks were almost unaffected. The experimental results demonstrate that the data augmentation method is effective for low-resource languages in multilingual dependency parsing. Furthermore, our experiments confirm that continuously increasing the quantity of synthetic data enhances UDify's performance. This improvement was particularly effective for zero-shot target languages.

Key Words: *Dependency Parsing, Multilingual Processing, Low-Resource Languages, Data Augmentation, Unsupervised Learning*

1 Introduction

Dependency parsers are tools used in computational linguistics to analyze the grammatical structure of sentences. They identify the relationships between words and highlight the dependencies that define grammatical meaning. Such insights are crucial for understanding syntactic roles and relationships within sentences and are essential for various language processing tasks. Efficiently training dependency parsers requires large treebanks (Dozat and Manning 2017; Qi et al. 2020; Straka and Straková 2020), which are structured datasets that annotate grammatical dependencies between words for specific languages. However, for low-resource languages that

[a] Gifu University
[b] National Institute of Information and Communications Technology (NICT)
A part of this work was done during the first author's internship at National Institute of Information and Communications Technology, Kyoto, Japan. Parts of this paper were published at PACLIC 2023 (Mao et al. 2023) and MWE-UD 2024 (Mao et al. 2024).

have no (zero-shot) or limited (few-shot) data available in treebanks, the challenge of training effective dependency parsers becomes significant. Multilingual modeling has emerged as an efficient solution in these cases, leveraging cross-lingual information to compensate for the lack of data on specific languages. Studies have demonstrated that performance on multilingual tasks can be boosted by pairing languages with similarities (Scholivet et al. 2019; Üstün et al. 2022). Furthermore, multilingualism also reduces the expense of training multiple models for a group of languages (Johnson et al. 2017; Aharoni et al. 2019; Muennighoff et al. 2023).

UDify (Kondratyuk and Straka 2019) exemplifies the effectiveness of this multilingual modeling approach. A multi-task model fine-tuned on multilingual BERT embeddings (Devlin et al. 2019), UDify has been trained using Universal Dependencies (UD) v2.3 (Zeman et al. 2018), which enables it to generate annotations for any treebank included in v2.3. It exhibits strong and consistent performance across all 124 UD treebanks for 75 languages, efficiently handling tasks such as part-of-speech (POS) tagging, lemmatization and dependency parsing. Despite its robust performance, an overlooked issue is the early saturation problem of the model in zero-shot languages, particularly evident in dependency parsing tasks, as illustrated in Figure 1, which shows the training process unlabeled attachment score (UAS) for example high-resource, few-shot, and zero-shot languages. In this figure, high-resource and few-shot languages demonstrate stable growth, while zero-shot languages show a gradual decline. This causes substantial discrepancies in the performance of methods such as UDify in zero-shot language scenarios, even when using almost identical training strategies, datasets, models, and evaluation methods, as reported in Üstün et al. (2020), Choudhary (2021), Üstün et al. (2022), Effland and Collins (2023).

While data augmentation methods have proven effective for high-resource languages (El-Kurdi

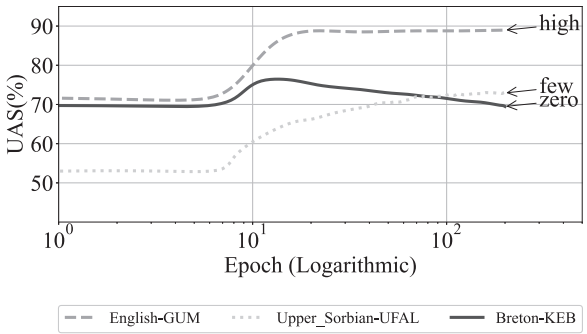

Figure 1 UAS performance trends during training for the `English-GUM` (high-resource), `Upper Sorbian-UFAL` (few-shot), and `Breton-KEB` (zero-shot) treebanks.

et al. 2020), they typically rely on existing annotated data and are thus not applicable to zero-shot languages, which lack such resources. To address the early saturation of dependency parsing in zero-shot languages, we propose a novel data augmentation strategy tailored specifically for these languages. Our approach begins by employing the trained UDify model to parse raw sentences from zero-shot languages, obtaining initial probabilities for dependency arcs. We then apply unsupervised learning techniques to refine these probabilities without the need for annotated data. The refined probabilities are then used to generate synthetic structured dependency data for the zero-shot languages. By integrating this synthetic data into UDify's training set, we expand the dataset's diversity and size, thereby enhancing UDify's performance and stability on zero-shot languages.

In this work, we conducted comprehensive experiments on low-resource languages using data augmentation methods, expanding (for few-shot languages) and creating (for zero-shot languages) synthetic treebanks for five few-shot and three zero-shot languages. By combining these synthetic treebanks with the UD treebanks and using the UDify framework, we trained a multilingual parser. As a result, increases in the UAS for zero-shot languages in the UD v2.13 were observed, with the average value increasing from 55.6% to 59.0%. In the best case, the UAS increased substantially from 52.7% to 61.0%. Similarly, few-shot languages experienced a UAS increase of 0.2%. By contrast, the UAS results for other languages and evaluation scores for other tasks did not show significant negative changes, suggesting that the overall robustness of the multilingual and multi-task processing was retained.

The remainder of this paper is organized as follows. In Section 2, we first introduce UD, UDify, and unsupervised dependency learning algorithms as the background of this study. Section 3 describes the performance of UDify on low-resource language scenarios, particularly on zero-shot languages, as well as a data augmentation method based on UDify and unsupervised learning algorithms. The experimental results of the case study are provided in Section 4. In addition, we discussed the impact of synthetic data quantity on the model and the enhancements made to single language data synthesis in Section 5. Section 6 concludes the paper and outlines future directions for this research.

2 Background

2.1 Universal Dependencies

Universal Dependencies (UD) are initiatives aimed at creating consistently annotated treebanks for numerous languages. Contributors from around the world have supplied treebanks for

hundreds of languages, thereby enhancing global linguistic analysis and comparison globally.

Some languages in the UD are represented by multiple treebanks. These treebanks are named using a structured format that combines the language name with an identifier for the data source or the creators. For example, the name `English-GUM` (Zeldes 2017) consists of `English` denoting the language and `GUM` indicating the Georgetown University Multilayer corpus project.[1] This systematic naming method ensures clarity and facilitates easy identification and comparison across different languages and sources.

We classify languages into high-resource and low-resource groups based on the available quantity of annotated data in the UD treebanks. High-resource languages have extensive annotated data, providing robust resources for linguistic analysis and computational tasks. Low-resource languages are further refined by Yang et al. (2022) into two groups: few-shot languages, which have only a small amount of training data from the UD treebanks, and zero-shot languages, which lack any training data and exist only in test datasets. This categorization facilitates a more systematic examination of UDify's performance across different linguistic resources, thereby providing crucial insights that guide our research.

Each treebank in the UD encompasses a rich array of grammatical annotations that characterize the linguistic features of sentences (Nivre et al. 2007). These include POS, which categorizes words according to their grammatical roles, lemmas that provide the base forms of words, morphological features describing inflectional properties, and dependency structures that map out the relationships between words. Figure 2 illustrates the directional arcs with relationships between words, with a key focus on the arcs in this study. UAS is a key evaluation metric to measure the accuracy of these arcs.

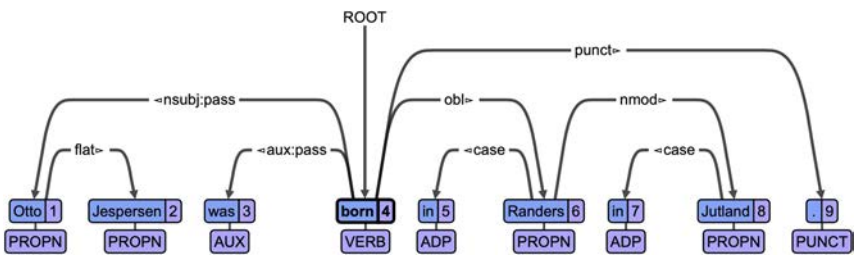

Figure 2 Visualization of "Otto Jespersen was born in Randers in Jutland." with POS tags, dependency arcs and relations from the `English-GUM` test set, using UD Annotatrix (Tyers et al. 2018).

[1] `gucorpling.org/gum/index.html`

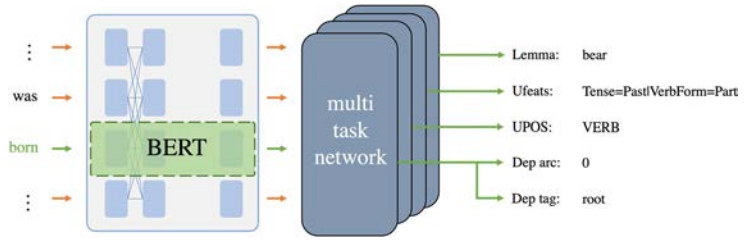

Figure 3 UDify model structure, which has a task-specific layer that processes word tokens and delivers UD annotations for each token, using the example from Figure 2. This figure is a redrawn adaptation based on illustrations from Kondratyuk and Straka (2019).

2.2 UDify

UDify is composed of a pre-trained mBERT model, which is a self-attention network with 12 transformer encoder layers (Vaswani et al. 2017), and a multi-task network for processing POS tags, lemmas, morphological features, and dependency structures, as illustrated in Figure 3. Unlike traditional models that require training specifically for each language, UDify uses mBERT's subword tokenization mechanism to jointly train across all UD treebanks. Note that for words tokenized into multiple subword units, only the first subword unit is input into the neural network designed for specific tasks.

The main focus of this study is dependency parsing, for which UDify uses a graph-based biaffine attention classifier (Dozat and Manning 2017). This classifier processes embeddings through specific neural layers, generating probability distributions that are crucial for accurate dependency parsing. The resulting dependency structures are decoded using the Chu–Liu/Edmonds algorithm (Chu 1965; Edmonds 1967). To optimize its use with mBERT for multilingual tasks, UDify is fine-tuned using an adapted version of the ULMFiT strategy (Howard and Ruder 2018), specifically modified for seamless integration with mBERT parameters.

Integrating these robust strategies enables UDify to excel in UD tasks across a variety of languages. Its effectiveness is especially notable in contexts involving dependency parsing for low-resource languages, where it delivers a reliable performance despite the limited data availability.

2.3 Unsupervised Dependency Learning

Using 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 optimal dependency arcs using a dynamic programming method with a CKY table based on the concepts of complete-link and

complete-sequence non-constituents. 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 on the bi-gram case in this study.

In the case of bi-grams, the directionality of a complete-link is determined by the outermost dependency direction, where $(w_i \rightarrow w_j)$ indicates a rightward link and $(w_i \leftarrow w_j)$ indicates a leftward link. The 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.

Unsupervised learning first constructs complete-links and complete-sequences for a substring, then incrementally merges the complete-links into larger complete-sequences and complete-sequences into larger complete-links. The recursion can be 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, while r and l indicate the direction (right or left). Furthermore, i and j represent the starting and ending indices of a word sequence in the sentence. This establishes the structural framework of the Unsupervised-Dep for complete-links and complete-sequences. The subsequent step applies the Inside–Outside algorithm (Lari and Young 1990; Lee and Choi 1997) to quantitatively determine the probabilities associated with each link and sequence. We divide the computational process of the Inside–Outside algorithm into three parts: `Inside`, `Outside`, and `Probability Updates`. The details of this process are provided in Appendix A.

During the computation of Unsupervised-Dep across the entire corpus, the initial random probabilities $P(w_i \rightarrow w_j)$ and $P(w_i \leftarrow w_j)$ are continuously trained and refined. Ultimately, this process results in the refined probabilities $P^{'}(w_i \rightarrow w_j)$ and $P^{'}(w_i \leftarrow w_j)$. The Viterbi algorithm (Forney 1973) determines the most probable tree structure within the inside portion of the input sentence $sent(1,n)$, generating the following optimal dependency arcs:

$$DEP_{arc} = Viterbi(CKY_{Inside}(sent(1,n), P^{'}(w_i \rightarrow w_j), P^{'}(w_i \leftarrow w_j))) \quad (1)$$

Here, CKY_{Inside} represents the results in the CKY table obtained from the inside portion of the calculations. Using the above method, it is possible to learn from the corpus of a specific language and generate a dependency treebank containing only dependency arcs without human intervention.

3 Proposed Method

3.1 Motivation

To ensure a comprehensive analysis, we initially used the UD v2.3 based on the original UDify training setup. However, recognizing the outdated nature of the UD v2.3, and ensuring the relevance and applicability of our findings, we incorporated the latest UD v2.13 treebanks into our study. This dual-dataset approach is crucial to rigorously evaluating the consistency of the early saturation phenomenon across different versions, affirming that our conclusions are robust and widely applicable.

Despite the original recommendation of 80 training epochs for UDify, our reproduction studies across both treebanks reveal that achieving results comparable to those reported in the literature typically required extending the training to approximately 200 epochs.[2] This extension is crucial not only for verifying the robustness of the early saturation effect but also for observing performance changes in few- and zero-shot languages as defined in Section 2.1. These changes are further analyzed in the subsequent sections.

Few-Shot Languages UDify demonstrates robust performance even on few-shot languages not included in the mBERT training set. For example, the Upper Sorbian language, which is absent from mBERT's training corpora, is notably under-resourced, with only 23 sentences in its `Upper_Sorbian-UFAL` treebank training set. Despite these limitations, UDify achieves impressive parsing accuracy in Upper Sorbian. The training process shows a consistent increase in parsing accuracy on the test set, as depicted in Figure 4.

Zero-Shot Languages For UD v2.3, the treebanks of 13 languages used for training UDify lack both training and development datasets. In contrast, for UD v2.13, there are 65 zero-shot languages. Notably, three of these languages—Breton, Tagalog, and Yoruba—are included in mBERT's training data. These languages have zero-shot treebanks in both UD v2.3 and v2.13, and UDify learns their dependency structures through transfer learning. Unlike high-resource and few-shot languages, the dependency parsing accuracy for zero-shot languages tends to decline

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

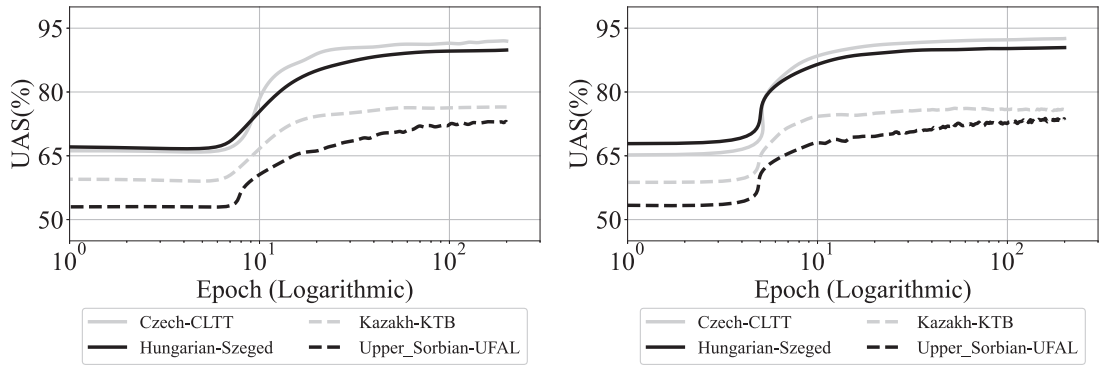

Figure 4 UAS performance trends during training for several few-shot languages: (left) results on UD v2.3, (right) results on UD v2.13.

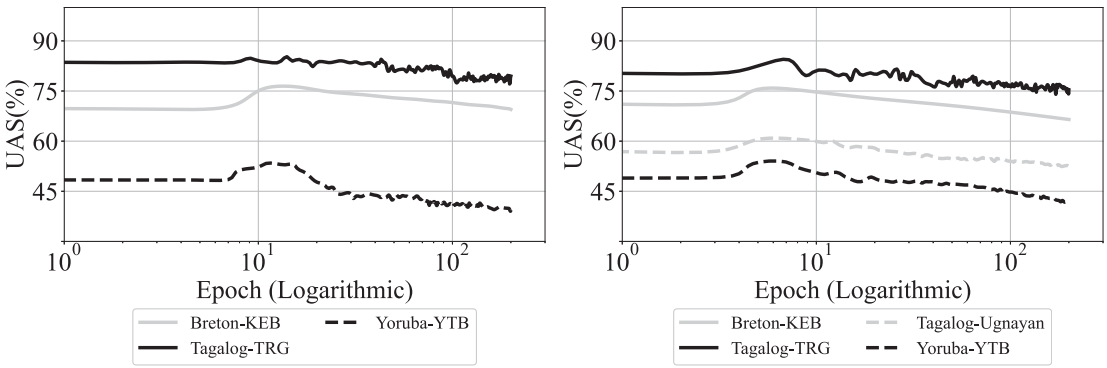

Figure 5 UAS performance trends during training for zero-shot languages: (left) results for UD v2.3, (right) results for UD v2.13. Note that `Tagalog-Ugnayan` appears only in UD v2.13.

after the initial few epochs of training. The optimal performance for any version of the treebank is typically observed around the 8th epoch, as depicted in Figure 5.

While the literature does not explicitly report a decrease in accuracy for zero-shot languages with extended training, our detailed analysis of results from multiple studies (Üstün et al. 2020; Choudhary 2021; Langedijk et al. 2022; Üstün et al. 2022) suggests such a trend. Given these disparities in training outcomes, where high-resource and few-shot languages show gradual improvements in contrast to the significant drops experienced by zero-shot languages, there is a pressing need for tailored strategies.

To address this issue, we have developed a data augmentation technique using Unsupervised-Dep specifically designed for multilingual parsers. This method is particularly crucial for zero-shot languages, as it extends their learning curve and stabilizes performance within multilingual

parsers. Additionally, our decision to include few-shot languages is motivated by the potential of data augmentation to enhance performance on specific tasks for languages with limited data resources (Ding et al. 2020).

This strategy involves enriching the training dataset with refined dependency structures generated through Unsupervised-Dep. The approach is designed not only to mitigate the early saturation problem in zero-shot languages but also to enhance performance consistency across both few- and zero-shot languages. By improving the stability and effectiveness of UDify's dependency parsing capabilities, this technique addresses the immediate challenges and broadens the applicability and impact of our method. Furthermore, it has the potential to lead to significant improvements in dependency parsing accuracy for both few- and zero-shot languages.

3.2 UDify with Unsupervised Augmentation

When applying the Unsupervised-Dep method to augment the data of UD treebanks, it is crucial to ensure that the generated data conform to the UD format. This approach, which we refer to as "Unsupervised Augmentation," requires not only generating dependency arcs but also producing other types of data. These additional data types must then be integrated with the results from Unsupervised-Dep. We divide the Unsupervised Augmentation into two main components, training and generation, and provide a detailed explanation of each.

Training Unsupervised-Dep has a high time complexity of $O(n^3)$, which makes the common practice in the original methods that start training from a random probability somewhat inefficient. To circumvent this, we 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 for a specific language *lang*, denoted as D_{train}^{lang}, into *trained-UDify* and generating dependency arcs represented by DEP_{arc}^{lang}. Subsequently, statistical computations are performed specifically on the DEP_{arc}^{lang} elements. The results, $P(w_i \rightarrow w_j)^{lang}$ and $P(w_i \leftarrow w_j)^{lang}$, serve as the initial probabilities for Unsupervised-Dep. Note that although the UDify parsing results include other types of information, such as POS, lemma, and dependecy relations (denoted as *other*), these are disregarded in the parsing results of D_{train}^{lang} because they do not participate in our subsequent computations, as depicted in Figure 6.

The input for Unsupervised-Dep consists of D_{train}^{lang} and the initial probabilities $P(w_i \rightarrow w_j)^{lang}$ and $P(w_i \leftarrow w_j)^{lang}$. After undergoing a set number of iterations, we obtain the

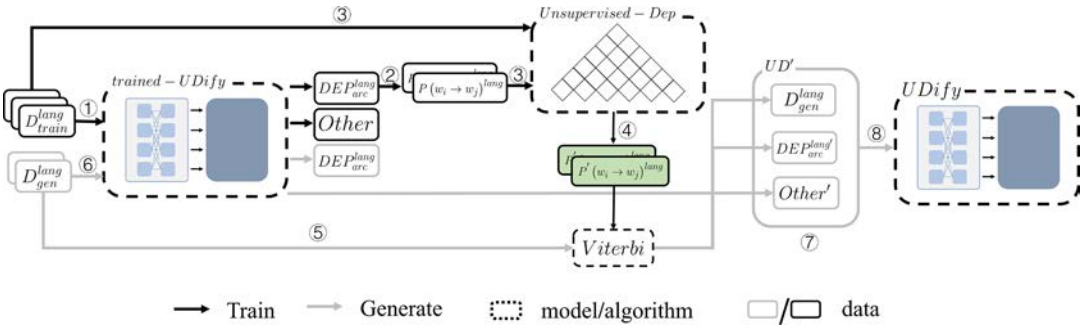

Figure 6 Integration of the parsing results from UDify with the training outcomes from Unsupervised-Dep for new UDify model training. The UDify model is a simplified diagram of that shown in Figure 3.

Algorithm 1 Unsupervised-Dep in multilingual corpora

for each $lang \in$ raw corpora **do**
  Input: $D_{train}^{lang} = sent_1, ..., sent_n, P(w_i \rightarrow w_j)^{lang}, P(w_i \leftarrow w_j)^{lang}$
  Output: $P^{'}(w_i \rightarrow w_j)^{lang}, P^{'}(w_i \leftarrow w_j)^{lang}$
  for each $sent \in D_{train}^{lang}$ **do**
    for $i = 1$ to length(s) **do**
      for $j = 1$ to length(s) **do**
        $\alpha_r^{Link}(i,j), \alpha_l^{Link}(i,j) = Inside(i, j, P(w_i \rightarrow w_j)^{lang}, P(w_i \leftarrow w_j)^{lang})$
    for $i = 1$ to length(s) **do**
      for $j = 1$ to length(s) **do**
        $\beta_r^{Link}(i,j), \beta_l^{Link}(i,j) = Outside(i, j, P(w_i \rightarrow w_j)^{lang}, P(w_i \leftarrow w_j)^{lang})$
    for $i = 1$ to length(s) **do**
      for $j = 1$ to length(s) **do**
        $P^{'}(w_i \rightarrow w_j)^{lang} = Probabilities\ Update(\alpha_r^{Link}(i,j)\beta_r^{Link}(i,j))$
        $P^{'}(w_i \leftarrow w_j)^{lang} = Probabilities\ Update(\alpha_l^{Link}(i,j)\beta_l^{Alnk}(i,j))$

re-estimated probabilities $P^{'}(w_i \rightarrow w_j)^{lang}$ and $P^{'}(w_i \leftarrow w_j)^{lang}$. This process adheres to the principles of the expectation–maximization algorithm, as Algorithm 1 confirms.

Generation To construct optimal dependency arcs for a sentence $sent(1, n)$ from D_{gen}^{lang} in the raw corpora, we utilize the re-estimated probabilities $P^{'}(w_i \rightarrow w_j)^{lang}$ and $P^{'}(w_i \leftarrow w_j)^{lang}$. The $Viterbi$ algorithm is employed to traverse the internal calculation results, as outlined in Eq. (1).

Following this, we use *trained-UDify* to parse D_{gen}^{lang}. The parsing results, termed DEP_{arc}^{lang} and $other^{'}$, are meticulously preserved. These data are illustrated in Figure 6 and play a crucial role in constructing the new UD format data, denoted as $UD^{'}$.

Subsequently, $UD^{'}$, which incorporates synthetic data for zero-shot languages, is integrated with the existing UD treebanks. This integration is specifically designed to train a new instance of UDify optimized for these languages. Notably, when a dependency arc exists between a word and the root, the relationship of this arc must be modified to "root." Dependency relationships between other words as parsed by *trained-UDify* remain unchanged. This selective adjustment ensures compliance with the "single root principle" of the dependency structure, which dictates that every sentence must have exactly one root, as cited in (Robinson 1970). Modifying the dependency relationship to root for arcs that connect to the "root" reinforces this principle, maintaining the structural integrity necessary for clear syntactic analysis.

The Unsupervised Augmentation approach not only generates synthetic data for zero-shot languages but also transitions them from relying solely on transfer learning to a blend of supervised and transfer learning. Consequently, this significantly enhances the performance of the multilingual model in zero-shot languages.

4 Experiments

4.1 Dataset

In this experiment, the data were divided into two categories: treebank data for training UDify and raw data for Unsupervised Augmentation. To validate the effectiveness of our method, we conducted experiments using both UD v2.3 and v2.13 treebanks. The two versions are compared in Table 1. Notably, UD v2.13 includes significantly more data than UD v2.3, which is the original version of the UD treebank used by UDify. Specifically, v2.13 has added 72 languages, of which 51 are zero-shot languages, further emphasizing the necessity of this study.

		UD v2.3	UD v2.13
Sentence	Train	745,860	1,414,197
	Dev	96,683	178,562
	Test	136,128	260,507
Tokens		18,001,694	30,817,267
Words		18,362,118	31,448,898
Language		76 (14)	148 (65)
Treebank		129 (35)	259 (107)

Table 1 Comparison of different versions of treebanks, where the numbers in parentheses indicate the number of languages/treebanks for which only test data are provided.

For the low-resource languages used in Unsupervised Augmentation, we selected eight languages that can be primarily tokenized using spaces, and we have detailed the analysis of their similarity to other high-resource languages in Appendix B. While our methodology is applicable to non-space-tokenized languages, we prioritized languages that facilitate a focused analysis on dependency structures without the complexities of tokenization, and are present in both versions of the UD treebanks. These include five few-shot languages and three zero-shot languages. We exhaustively collected data for these languages from publicly available corpora (Tiedemann 2012; El-Kishky et al. 2020; Reimers and Gurevych 2020; Schwenk et al. 2021), and divided the raw data for each language into parts D_{train} and D_{gen}. We set D_{gen} to 1,000, a decision that sets the stage for exploration of synthetic data quantity across multiple low-resource languages and its impact on UAS in subsequent experiments. The details and statistics of the collected data are organized and presented in Table 2 and referred to as "OPUS-mult" in subsequent sections.

Language	$\#D^{lang}_{train}$	$\#D^{lang}_{gen}$	Treebank	#train	#test	Code
Hungarian	134.1	1,000	Hungarian-Szeged	910	449	hu
Kazakh	1.7	1,000	Kazakh-KTB	31	1,047	kk
Lithuanian	236.7	1,000	Lithuanian-ALKSNIS^{+}	2,341	684	lt-a
			Lithuanian-HSE	153	55	lt-h
Marathi	1.5	1,000	Marathi-UFAL	373	47	mr
Tamil	13.7	1,000	Tamil-MWTT^{+}	0	534	ta-m
			Tamil-TTB	400	120	ta-t
Breton	18.2	1,000	Breton-KEB*	—	884	br
Tagalog	150.0	1,000	Tagalog-TRG*	—	128	tl-t
			Tagalog-Ugnayan^{+}	—	94	tl-u
Yoruba	9.7	1,000	Yoruba-YTB*	—	318	yo

Table 2 Raw data collected from various corpora and the treebanks of the collected languages in UD v2.13. The top five languages are few-shot languages and the bottom three languages are the zero-shot languages. $\#D^{lang}_{train}$ indicates the number of raw sentences (in thousands) used for training in Unsupervised Augmentation, and $\#D^{lang}_{gen}$ indicates the raw sentences used for generating in Unsupervised Augmentation. #train and #test are the number of sentences in the training and test sets, respectively. The code represents the abbreviated name of the treebank, facilitating the display of the experimental results in subsequent sections. Treebanks marked with an "*" indicate that they exist in both versions, but those in UD v2.13 have with changes including but not limited to the number of test sets and error corrections. Treebanks marked with a "+" indicates that they are exclusive to UD v2.13.

4.2 Setup

To minimize the impact of potential experimental environment changes, as discussed in Popel and Bojar (2018), we adhered to the parameter settings outlined in Kondratyuk and Straka (2019) and reimplemented the model as UDify(our) across different versions of the treebanks. Additionally, to expedite the experimental process, we employed Horovod (Sergeev and Balso 2018) to modify UDify and AllenNLP (Gardner et al. 2018) for multi-GPU parallel training.[3] During the training of the Unsupervised Augmentation, we used the UDify(our) model to parse each language present in the OPUS-mult dataset. The statistical results derived from the parsing outcomes of each language were adopted as their initial probabilities, which were continuously re-estimated throughout the unsupervised learning process. After the 10th training iteration, we used the newly estimated probabilities to parse sentences from the OPUS-mult dataset, generating data that align with the UD format.

To explore the impact of expanding and creating training data for low-resource languages on parsing accuracy, we conducted methodological experiments. In these experiments, we randomly selected 300 sentences for each language from D_{gen} in OPUS-mult, processed them through the generation part of Unsupervised Augmentation, and integrated them into the UD treebanks to form the "Unsup" training dataset. Inspired by the work of Triguero et al. (2015), Rybak and Wróblewska (2018), we conducted an experiment using a comparative method called "Self." In this approach, we used the same raw sentences as Unsup and applied the parsing results from the UDify(org) model. These results were merged with the original training set to train the Self model.

During training, following McDonald et al. (2011), we merged training sets, randomized the sentence order in each epoch, and fed the network diverse batches of original and synthetic data from multiple languages. Additionally, we controlled the amount of data in each epoch to ensure that each experiment used the same quantity of training data.

4.3 Result

A comparison with the experimental results reported in Kondratyuk and Straka (2019) confirms that UDify(our) was successfully reimplemented, as shown in Table 3. Although no single method yielded significant improvements for few-shot languages, the results for zero-shot languages, analyzed using bootstrap resampling (Efron 1992), demonstrate a consistent and significant enhancement in UDify's dependency arc accuracy through the Unsup strategy during

[3] When replicated across 16 V100s in parallel, replicating UDify takes close to 9 days on UD v2.3 and close to 18 days on UD v2.13.

	Code	v2.3				v2.13		
		UDify(org)	UDify(our)	Self	Unsup	UDify(our)	Self	Unsup
Few-shot	hu	**89.7**	89.8	89.7	89.6	90.4	**90.6**	90.5
	kk	74.8	76.3	76.0	76.3	75.8	75.1	**75.9**
	lt-a	—	—	—	—	82.8	**83.2**	83.1
	lt-h	79.1	79.1	79.1	78.9	**81.2**	79.7	80.9
	mr	**79.4**	73.1	74.8	74.3	72.7	73.5	**73.8**†
	ta-m	—	—	—	—	83.0	83.3	**83.4**
	ta-t	79.3	80.2	**80.6**	80.0	78.2	**78.3**	78.1
Zero-shot	br	63.5	69.7	69.1	**73.4**‡	66.3	66.0	**70.0**‡
	tl-t	64.0	79.5	75.7	**87.3**‡	75.3	80.1	**82.6**‡
	tl-u	—	—	—	—	52.7	53.7	**61.0**‡
	yo	37.6	39.4	40.2	**41.3**	41.3	41.2	**43.1**†
Few		—	**82.6**	82.4	82.5	82.6	82.7	**82.8**
Zero		—	63.8	63.3	**67.2**‡	55.6	55.6	**59.0**‡

Table 3 UAS (%) for few- and zero-shot languages obtained using different methods on UD v2.3 and v2.13. The last two rows display the combined test set results for few- (Few) and for zero-shot (Zero) languages. We denote the treebank names using the codes from Table 1. The UDify(org) result for the v2.3 was reported in Kondratyuk and Straka (2019). The best result in each group is highlighted in **bold**. Additionally, we conducted significance tests on the experimental results, comparing UDify(our) with Unsup, where "‡" indicates a significance level of $p \leq 0.01$ and "†" indicates $p \leq 0.05$, both showing where Unsup outperforms UDify(our).

training on different versions of the treebanks. This is reflected in the combined test set results, where the UAS for UD v2.3 increases from 63.8% to 67.2% and the UAS for UD v2.13 increases from 55.6% to 59.0%. Since our study primarily focuses on the parsing accuracy of dependency arcs (measured by UAS), and given that the LAS has also shown improvement, we have included the LAS results in Appendix C.

To the best of our knowledge, this is the state-of-the-art result for Tagalog (Aquino and de Leon 2020; Choudhary 2021). In UD v2.13, the UAS for treebanks `Tagalog-TRG` and `Tagalog-Ugnayan` showed remarkable increases from 75.3% and 52.7% to 82.6% and 61.0%. Using Breton from the zero-shot languages in UD v2.3 and v2.13 as an example, we illustrate the changes in UAS during the training process under different methods in Figure 7. The figure reveals that incorporating data generated through Unsupervised Augmentation significantly mitigates the decline in UAS accuracy during the training of zero-shot languages, thereby improving outcomes. In the experimental results for UD v2.13, the performance of Unsup was better than that of UDify(our), but it still remains significantly lower than the accuracy at the start of training.

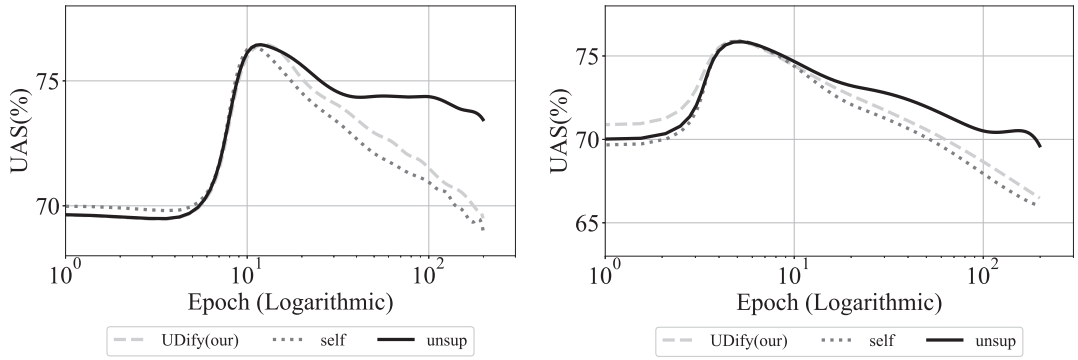

Figure 7 UAS performance trends during training are shown for the `Breton-KEB` test set, comparing two baselines, UDify(our) and Self, with the proposed method, Unsup. (Left) Results for UD v2.3, (right) results for UD v2.13.

Given that UDify is a multilingual dependency parser, it is essential to assess the impact of our proposed method on other languages. To further investigate the differences in UAS results between UDify(org) and Unsup, we tested all treebanks in UD v2.3 and v2.13. As shown in Figure 8, the results indicate that using Unsupervised Augmentation to create synthetic data effectively enhances the UAS for zero-shot languages, particularly for Breton and Tagalog. However, considering that UD v2.13 contains a larger proportion of zero-shot treebanks (approximately 41.3%), the Unsup method shows unstable results on these treebanks. This includes instances in which the UAS significantly increased for certain languages without data augmentation. Despite these fluctuations, the method's negative impact on the parsing precision of dependency structures in other languages remains minimal.

For a comprehensive comparison, the UD scores of the zero-shot and other languages have been compiled in Table 4.[4] Although our primary research objective was to improve the parsing accuracy of dependency arcs, we have also enhanced UDify's performance in other tasks for low-resource languages. UDify must balance the loss produced by multiple decoders during training, and as noted in the work of Rybak and Wróblewska (2018), these variations in evaluation metrics are considered reasonable. Broadly, our method does not negatively impact other languages or tasks, maintaining their performance levels.

Considering the current results, we believe that using Unsupervised Augmentation to synthesize training data for zero-shot languages is necessary and effective in a multilingual modeling

[4] Except for `UD_Chukchi-HSE`, the format of this treebank is not supported by the script used to calculate the UD scores.

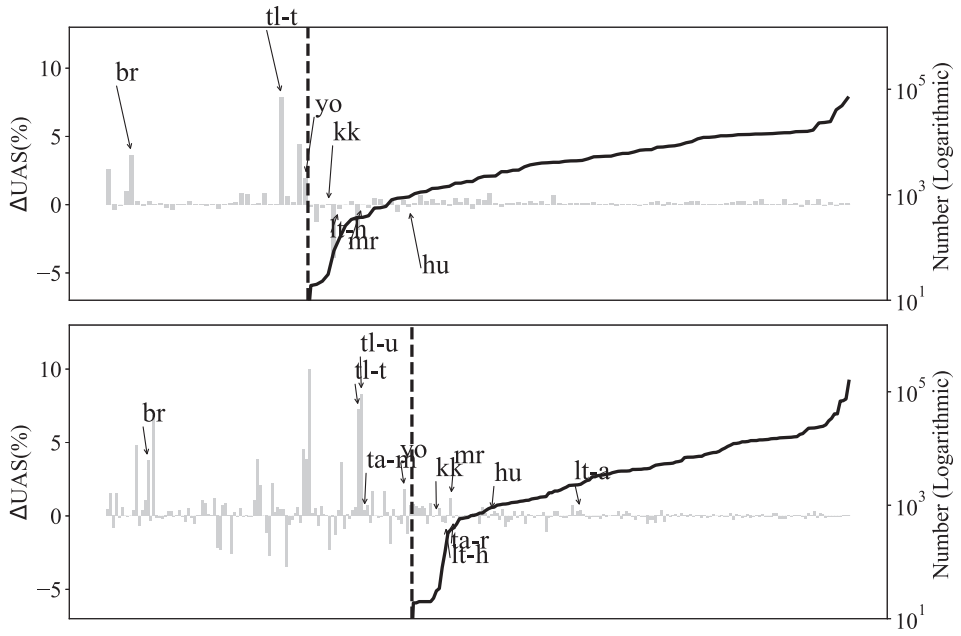

Figure 8 Difference in UAS (%) across the test treebanks UD v2.3 (top) and UD v2.13 (bottom). The gray bars above the X-axis indicate Unsup > UDify(our), and those below the X-axis indicate Unsup < UDify(our) (left Y-axis). The black line (right Y-axis) shows treebank training set sizes. Zero-shot languages are to the left of the black dashed line. Note that because Tamil is represented by both treebanks `Tamil-MWTT` (ta-m) and `Tamil-TTB` (ta-t), we classify ta-m as few-shot.

	v2.3				v2.13			
	Zero-shot		Other		Zero-shot		Other	
	UAS	Rest	UAS	Rest	UAS	Rest	UAS	Rest
UDify(our)	63.8	55.0	77.5	82.4	55.6	56.0	80.1	83.6
Self	63.3	58.7	77.6	82.5	55.6	56.4	80.1	83.5
Unsup	**67.2**	**61.4**	77.6	82.5	**59.0**	**59.3**	80.2	83.6

Table 4 UD scores on selected zero-shot and other languages obtained by different methods. Rest(%) refers to the average scores of UPOS, UFeats, Lemmas, and LAS in the UD scores. The best result in each group is in highlighted in **bold**.

context. In the following sections, we focus on UD v2.13 to discuss how the number of zero-shot languages synthesized by Unsupervised Augmentation impacts the UAS and its performance when applied to a single language exclusively.

5 Discussion

5.1 Quantity of Synthetic Data in Multiple Low-Resource Languages

To further investigate the effectiveness of Unsupervised Augmentation, we conducted experiments on the impact of generating different quantities of synthetic data for low-resource languages on UAS. These experiments were carried out using UD v2.13. Maintaining the same experimental setup and parameters as in the previous chapter, we set a fixed amount of data for the input per epoch. To ensure the same number of model updates in each epoch and to avoid variations in the final UAS values due to update frequencies differences, we randomly discarded some data during each epoch. We set the number of synthetic data points to 100, 300, 500, and 1,000, representing sentences randomly selected from D_{gen}, producing a total of 800, 2,400, 4,000, and 8,000 synthetic data points for each of the eight languages listed in Table 2, named Unsup100, Unsup300, Unsup500, and Unsup1000. The results of the experiments are presented in Table 5, and the specific changes in UAS for zero-shot languages during the training process are illustrated in Figures 9 and 10.

The experimental results and observed changes in UAS during the training process reveal that increasing the synthetic data quantity enhances both the stability during training and the final outcomes. Additionally, increasing synthetic data quantity can mitigate or suppress the downward trends observed during training. Specifically for `Tagalog-TRG` and `Tagalog-Ugnayan`, setting the synthetic data quantity to 1k significantly suppresses this downward trend. Although a decrease is still observed in `Breton-KEB` and `Yoruba-YTB`, the final results are superior to those obtained with smaller amounts of synthetic data. Moreover, for `Breton-KEB`, increasing the synthetic data quantity to 500 resulted in final UAS values that surpass those at the start of training. The performance metrics for selected zero-shot languages and other data not using Unsupervised Augmentation in UD tasks are shown in Table 6, with no significant impact observed on other languages or tasks.

	Few-shot							Zero-shot				Few	Zero
	hu	kk	lt-a	lt-h	mr	ta-m	ta-t	br	tl-t	tl-u	yo		
Unsup_{100}	90.4	75.6	82.9	80.8	73.8	**83.7**	**79.1**	66.8	83.0	57.4	43.1	82.7	58.2
Unsup_{300}	**90.5**	75.9	**83.1**	80.8	73.8	83.4	78.1	70.0	82.6	61.0	43.1	**82.8**	59.0
Unsup_{500}	90.3	75.6	83.0	**82.1**	**74.3**	83.0	78.6	70.7	84.3	61.0	**44.8**	82.7	60.1
Unsup_{1000}	90.4	**76.0**	82.9	81.4	73.1	82.2	78.9	**71.7**	**86.4**	**62.9**	44.7	82.7	**60.7**

Table 5 UAS for few- and zero-shot languages obtained using different quantities of synthetic data. The best result in each group is in highlighted in **bold**.

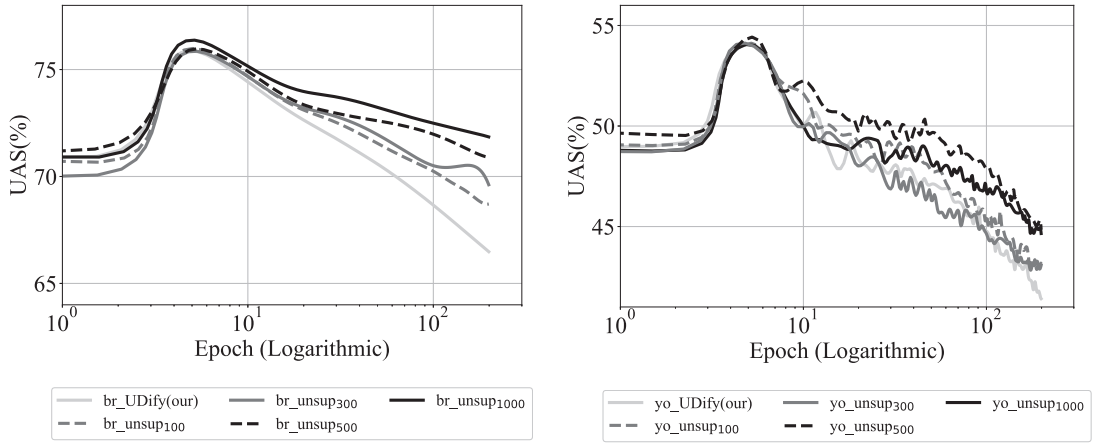

Figure 9 UAS performance trends during training: (left) results for `Breton-KEB`, (right) results for `Yoruba-YTB`.

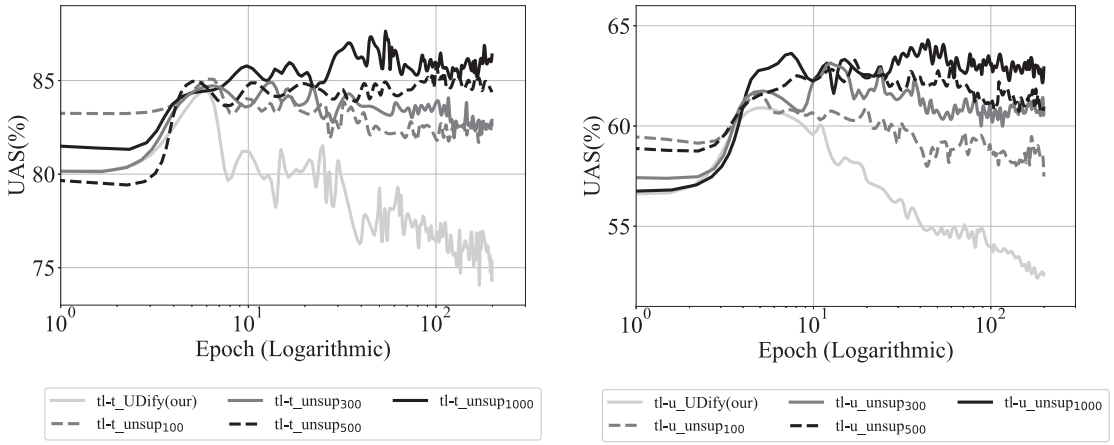

Figure 10 UAS performance trends during training: (left) results for `Tagalog-TRG`, (right) results for `Tagalog-Ugnayan`.

5.2 Quantity of Synthetic Data in a Single Low-Resource Language

When data augmentation is simultaneously performed on multilingual language models, it may lead to unexpected effects on other languages (Wang et al. 2020). Considering the benefits of targeted data augmentation in a multilingual environment (Choi et al. 2024), we conducted experiments focusing on data augmentation for a single language treebank. This experiment evaluated the specific improvements in model performance attributable to augmenting the data for one language at a time.

	Zero-shot		Other	
	UAS	Rest	UAS	Rest
Unsup_{100}	58.2	58.3	80.1	83.5
Unsup_{300}	59.0	59.3	80.1	83.6
Unsup_{500}	60.1	60.1	80.1	83.6
Unsup_{1000}	**60.7**	**61.0**	80.1	83.6

Table 6 UD scores on selected zero-shot and other languages obtained by different methods. Rest(%) indicates the average score of UPOS, UFeats, Lemma, and LAS in the UD scores. The best result in each group is in highlighted in **bold**.

	Zero-shot		Other	
	UAS	Rest	UAS	Rest
br_{100}	68.3	56.4	80.0	83.5
br_{300}	69.6	58.3	80.1	83.5
br_{500}	70.5	58.9	80.1	83.6
br_{1000}	**72.1**	**60.3**	80.1	83.5

Table 7 UD scores on `Breton-KEB` obtained by different amounts of synthetic data . Rest(%) indicates the average score of UPOS, UFeats, Lemma, and LAS in the UD scores. The best result in each group is in highlighted in **bold**.

Because `Breton-KEB` is the largest treebank of zero-shot language in our experiments, it allows for a more effective evaluation of Unsupervised Augmentation's performance on a single language. Therefore, we selected Breton as our target for Unsupervised Augmentation. Regarding the synthetic data quantity, we maintained the same parameters as in other experiments: 100, 300, 500, and 1,000, labeled as br_{100}, br_{300}, br_{500}, and br_{1000}, respectively. The results of these experiments are presented in Table 7, and the specific changes in UAS during the training process for the `Breton-KEB` are illustrated in Figure 11.

For a visual comparison, we plotted the changes during the training process using the best-performing parameters in Figure 11. The graph shows that, in a multilingual model scenario, whether synthetic data are produced for multiple languages simultaneously or only for a single language, the results are similar and do not exhibit significant changes, such as suppressing the decreasing UAS trend.

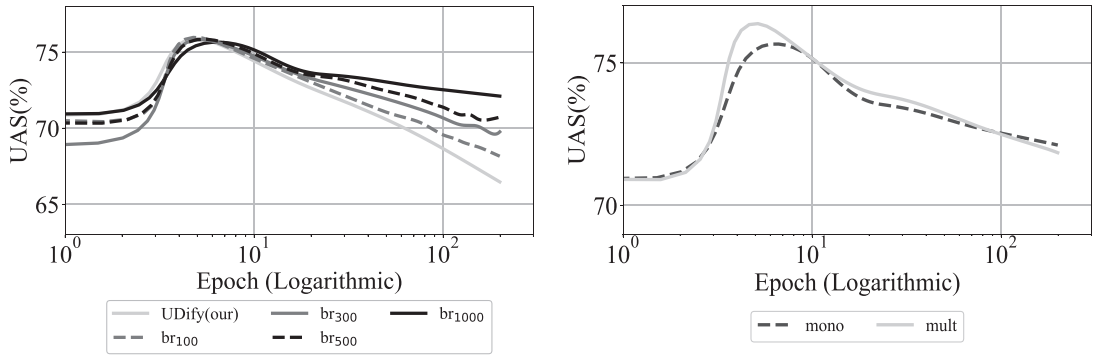

Figure 11 (Left) UAS performance trends during training for the `Breton-KEB` test set. (Right) When the synthetic data quantity is set to 1000, the changes in UAS during the training process for the `Breton-KEB` test set are observed, both when using Unsupervised Augmentation on multiple languages simultaneously and when applying it to a single language only.

6 Conclusion and Future Work

This study focuses on the phenomenon by which the dependency-parsing accuracy of UDify in low-resource language scenarios declines during the training process. We observed that maintaining the number of training epochs was essential to preserve the overall performance of the multilingual parser, as its accuracy on the entire test set continued to improve gradually with extended training. Recognizing the need to separately optimize the parsing capabilities of zero-shot languages within a multilingual parser, we propose a data augmentation method based on unsupervised learning, specifically tailored to enhance the final performance in zero-shot languages. The effectiveness of our approach in zero-shot languages was confirmed by multilingual experimental results, demonstrating significant improvements in dependency parsing accuracy without adversely affecting UDify's performance on other languages and tasks. Furthermore, we examined the impact of synthetic data quantity and the use of synthetic data for individual zero-shot languages. The experimental results indicate that increasing the quantity of synthetic data can effectively improve the UAS during training with UDify. However, using synthetic data exclusively for a single language does not significantly enhance its accuracy at the end of training.

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

Acknowledgement

This work builds upon the Introduction and Proposed Method sections of two papers accepted by *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation* (PACLIC 2023) (Mao et al. 2023) and *Proceedings of the Joint Workshop on Multiword Expressions and Universal Dependencies* (MWE-UD 2024) (Mao et al. 2024). We have carefully extended the background section of this study. For the experiments in Section 4, we primarily use UD v2.13. Additionally, in Section 5, we analyzed the impact of the synthetic data quantity on the experimental outcomes and a comparison of results between multilingual and monolingual applications. We would like to express our gratitude to the anonymous reviewers for their valuable comments and suggestions. This work was financially supported by JST SPRING, Grant Number JPMJSP2125. The author (Initial) would like to take this opportunity to thank the "THERS Make New Standards Program for the Next Generation Researchers."

References

Aharoni, R., Johnson, M., and Firat, O. (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)*, pp. 3874–3884, Minneapolis, Minnesota. Association for Computational Linguistics.

Aquino, A. and de Leon, F. (2020). "Parsing in the absence of related languages: Evaluating low-resource dependency parsers on Tagalog." In de Marneffe, M.-C., de Lhoneux, M., Nivre, J., and Schuster, S. (Eds.), *Proceedings of the 4th Workshop on Universal Dependencies (UDW 2020)*, pp. 8–15, Barcelona, Spain (Online). Association for Computational Linguistics.

Choi, C., Jeong, Y., Park, S., Won, I., Lim, H., Kim, S., Kang, Y., Yoon, C., Park, J., Lee, Y., Hahm, Y., Kim, H., and Lim, K. (2024). "Optimizing Language Augmentation for Multilingual Large Language Models: A Case Study on Korean." *arXiv preprint arXiv:2403.10882*.

Choudhary, C. (2021). "Improving the Performance of UDify with Linguistic Typology Knowledge." In *Proceedings of the 3rd Workshop on Computational Typology and Multilingual NLP*, pp. 38–60, Online. Association for Computational Linguistics.

Chronopoulou, A., Stojanovski, D., and Fraser, A. (2023). "Language-Family Adapters for Low-Resource Multilingual Neural Machine Translation." In Ojha, A. K., Liu, C.-h., Vylomova, E., Pirinen, F., Abbott, J., Washington, J., Oco, N., Malykh, V., Logacheva, V., and Zhao,

X. (Eds.), *Proceedings of the 6th Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2023)*, pp. 59–72, Dubrovnik, Croatia. Association for Computational Linguistics.

Chu, Y.-J. (1965). "On the Shortest Arborescence of A Directed Graph." *Scientia Sinica*, **14**, pp. 1396–1400.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (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)*, pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ding, B., Liu, L., Bing, L., Kruengkrai, C., Nguyen, T. H., Joty, S., Si, L., and Miao, C. (2020). "DAGA: Data Augmentation with a Generation Approach for Low-resource Tagging Tasks." In Webber, B., Cohn, T., He, Y., and Liu, Y. (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6045–6057, Online. Association for Computational Linguistics.

Ding, C. and Yamamoto, M. (2013). "An Unsupervised Parameter Estimation Algorithm for a Generative Dependency N-gram Language Model." In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pp. 516–524, Nagoya, Japan. Asian Federation of Natural Language Processing.

Ding, C. and Yamamoto, M. (2014). "A Generative Dependency N-gram Language Model: Unsupervised Parameter Estimation and Application." *Information and Media Technologies*, **9** (4), pp. 857–885.

Dozat, T. and Manning, C. D. (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.

Edmonds, J. (1967). "Optimum Branchings." *Journal of Research of the National Bureau of Standards B*, **71** (4), pp. 233–240.

Effland, T. and Collins, M. (2023). "Improving Low-Resource Cross-lingual Parsing with Expected Statistic Regularization." *Transactions of the Association for Computational Linguistics*, **11**, pp. 122–138.

Efron, B. (1992). "Bootstrap Methods: Another Look at the Jackknife." In *Breakthroughs in statistics: Methodology and distribution*, pp. 569–593. Springer.

El-Kishky, A., Chaudhary, V., Guzmán, F., and Koehn, P. (2020). "CCAligned: A Massive Collection of Cross-Lingual Web-Document Pairs." In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing (EMNLP), pp. 5960–5969, Online. Association for Computational Linguistics.

El-Kurdi, Y., Kanayama, H., Sarioglu Kayi, E., Castelli, V., Ward, T., and Florian, R. (2020). "Scalable Cross-lingual Treebank Synthesis for Improved Production Dependency Parsers." In Clifton, A. and Napoles, C. (Eds.), *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pp. 172–178, Online. International Committee on Computational Linguistics.

Forney, G. D. (1973). "The Viterbi Algorithm." *Proceedings of the IEEE*, **61** (3), pp. 268–278.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (2018). "AllenNLP: A Deep Semantic Natural Language Processing Platform." In Park, E. L., Hagiwara, M., Milajevs, D., and Tan, L. (Eds.), *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pp. 1–6, Melbourne, Australia. Association for Computational Linguistics.

Howard, J. and Ruder, S. (2018). "Universal Language Model Fine-tuning for Text Classification." In Gurevych, I. and Miyao, Y. (Eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, Melbourne, Australia. Association for Computational Linguistics.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation." *Transactions of the Association for Computational Linguistics*, **5**, pp. 339–351.

Kondratyuk, D. and Straka, M. (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)*, pp. 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Kong, X., Renduchintala, A., Cross, J., Tang, Y., Gu, J., and Li, X. (2021). "Multilingual Neural Machine Translation with Deep Encoder and Multiple Shallow Decoders." In Merlo, P., Tiedemann, J., and Tsarfaty, R. (Eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1613–1624, Online. Association for Computational Linguistics.

Langedijk, A., Dankers, V., Lippe, P., Bos, S., Cardenas Guevara, B., Yannakoudakis, H., and Shutova, E. (2022). "Meta-Learning for Fast Cross-Lingual Adaptation in Dependency Parsing." In Muresan, S., Nakov, P., and Villavicencio, A. (Eds.), *Proceedings of the 60th An-*

nual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8503–8520, Dublin, Ireland. Association for Computational Linguistics.

Lari, K. and Young, S. J. (1990). "The Estimation of Stochastic Context-free Grammars using the Inside-Outside Algorithm." *Computer Speech & Language*, **4** (1), pp. 35–56.

Lee, S. and Choi, K. (1997). "Reestimation and Best-First Parsing Algorithm for Probabilistic Dependency Grammars." In Zhou, J. and Church, K. (Eds.), *5th Workshop on Very Large Corpora, VLC 1997, Beijing, China and Hong Kong, August 18 and August 20, 1997*.

Littell, P., Mortensen, D. R., Lin, K., Kairis, K., Turner, C., and Levin, L. (2017). "Uriel and lang2vec: Representing Languages as Typological, Geographical, and Phylogenetic Vectors." In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Vol. 2, pp. 8–14.

Mao, J., Ding, C., Kaing, H., Tanaka, H., Utiyama, M., and Matsumoto, T. (2023). "Improving Zero-Shot Dependency Parsing by Unsupervised Learning." In Huang, C.-R., Harada, Y., Kim, J.-B., Chen, S., Hsu, Y.-Y., Chersoni, E., A, P., Zeng, W. H., Peng, B., Li, Y., and Li, J. (Eds.), *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pp. 217–226, Hong Kong, China. Association for Computational Linguistics.

Mao, J., Ding, C., Kaing, H., Tanaka, H., Utiyama, M., and Matsumoto., T. (2024). "Overcoming Early Saturation on Low-Resource Languages in Multilingual Dependency Parsing." In Bhatia, A., Bouma, G., Doğruöz, A. S., Evang, K., Garcia, M., Giouli, V., Han, L., Nivre, J., and Rademaker, A. (Eds.), *Proceedings of the Joint Workshop on Multiword Expressions and Universal Dependencies (MWE-UD) @ LREC-COLING 2024*, pp. 63–69, Torino, Italia. ELRA and ICCL.

McDonald, R., Petrov, S., and Hall, K. (2011). "Multi-Source Transfer of Delexicalized Dependency Parsers." In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Le Scao, T., Bari, M. S., Shen, S., Yong, Z. X., Schoelkopf, H., Tang, X., Radev, D., Aji, A. F., Almubarak, K., Albanie, S., Alyafeai, Z., Webson, A., Raff, E., and Raffel, C. (2023). "Crosslingual Generalization through Multitask Finetuning." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15991–16111, Toronto, Canada. Association for Computational Linguistics.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). "The CoNLL 2007 Shared Task on Dependency Parsing." In Eisner, J. (Ed.), *Proceedings*

of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 915–932, Prague, Czech Republic. Association for Computational Linguistics.

Popel, M. and Bojar, O. (2018). "Training Tips for the Transformer Model." *The Prague Bulletin of Mathematical Linguistics*, **110**, pp. 43–70.

Qi, P., Dozat, T., Zhang, Y., and Manning, C. D. (2018). "Universal Dependency Parsing from Scratch." In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pp. 160–170, Brussels, Belgium. Association for Computational Linguistics.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (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*, pp. 101–108, Online. Association for Computational Linguistics.

Reimers, N. and Gurevych, I. (2020). "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Robinson, J. J. (1970). "Dependency Structures and Transformational Rules." *Language*, **46** (2), pp. 259–285.

Rybak, P. and Wróblewska, A. (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*, pp. 45–54, Brussels, Belgium. Association for Computational Linguistics.

Scholivet, M., Dary, F., Nasr, A., Favre, B., and Ramisch, C. (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)*, pp. 3919–3930, Minneapolis, Minnesota. Association for Computational Linguistics.

Schwenk, H., Chaudhary, V., Sun, S., Gong, H., and Guzmán, F. (2021). "WikiMatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia." In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1351–1361, Online. Association for Computational Linguistics.

Sergeev, A. and Balso, M. D. (2018). "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow." *CoRR*, **abs/1802.05799**.

Straka, M. and Straková, J. (2020). "UDPipe at EvaLatin 2020: Contextualized Embeddings

and Treebank Embeddings." In Sprugnoli, R. and Passarotti, M. (Eds.), *Proceedings of LT4HALA 2020 - 1st Workshop on Language Technologies for Historical and Ancient Languages*, pp. 124–129, Marseille, France. European Language Resources Association (ELRA).

Tiedemann, J. (2012). "Parallel Data, Tools and Interfaces in OPUS." In Calzolari, N., Choukri, K., Declerck, T., Dogan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S. (Eds.), *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Tran, K. and Bisazza, A. (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)*, pp. 281–288, Hong Kong, China. Association for Computational Linguistics.

Triguero, I., García, S., and Herrera, F. (2015). "Self-labeled Techniques for Semi-supervised Learning: Taxonomy, Software and Empirical Study." *Knowledge and Information Systems*, **42** (2), pp. 245–284.

Tyers, F. M., Sheyanova, M., and Washington, J. N. (2018). "UD Annotatrix: An annotation tool for Universal Dependencies." In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16)*, pp. 10–17.

Üstün, A., Bisazza, A., Bouma, G., and van Noord, G. (2020). "UDapter: Language Adaptation for Truly Universal Dependency Parsing." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2302–2315, Online. Association for Computational Linguistics.

Üstün, A., Bisazza, A., Bouma, G., and van Noord, G. (2022). "UDapter: Typology-based Language Adapters for Multilingual Dependency Parsing and Sequence Labeling." *Computational Linguistics*, **48** (3), pp. 555–592.

van der Maaten, L. and Hinton, G. (2008). "Visualizing Data using t-SNE." *Journal of Machine Learning Research*, **9** (86), pp. 2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). "Attention is All you Need." In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

Wang, Z., Lipton, Z. C., and Tsvetkov, Y. (2020). "On Negative Interference in Multilingual Models: Findings and A Meta-Learning Treatment." In Webber, B., Cohn, T., He, Y., and Liu, Y. (Eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4438–4450, Online. Association for Computational Linguistics.

Yang, Z., Fang, Q., and Feng, Y. (2022). "Low-resource Neural Machine Translation with Cross-modal Alignment." In Goldberg, Y., Kozareva, Z., and Zhang, Y. (Eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10134–10146, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zeldes, A. (2017). "The GUM Corpus: Creating Multilayer Resources in the Classroom." *Language Resources and Evaluation*, **51** (3), pp. 581–612.

Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., and Petrov, S. (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*, pp. 1–21, Brussels, Belgium. Association for Computational Linguistics.

Appendix

A Inside–Outside Algorithm

The algorithm begins by calculating the inside probabilities from bottom to top and left to right across the CKY table, and then it computes the outside probabilities from top to bottom and right to left, utilizing the previously calculated inside probabilities.

Inside The inside probability of a complete-link reflects the likelihood of generating a word sequence $\mathrm{w}_{i,j}$ when there is a dependency relation between w_i and w_j as follows:

$$\begin{aligned} \alpha_r^{Link}(i,j) &= \textstyle\sum_{m=i}^{j-1} p(w_i \rightarrow w_j)\alpha_r^{Seq}(i,m)\alpha_r^{Seq}(m+1,j) \\ \alpha_l^{Link}(i,j) &= \textstyle\sum_{m=i}^{j-1} p(w_i \leftarrow w_j)\alpha_r^{Seq}(i,m)\alpha_l^{Seq}(m+1,j) \end{aligned} \tag{2}$$

The inside probability for a complete-sequence represents the likelihood that the word sequence $\mathrm{w}_{i,j}$ originates from either $Seq_l(i,j)$ or $Seq_r(i,j)$ as follows:

$$\begin{aligned} \alpha_r^{Seq}(i,j) &= \textstyle\sum_{m=i}^{j-1} \alpha_r^{Seq}(i,m)\alpha_r^{Link}(m,j) \\ \alpha_l^{Seq}(i,j) &= \textstyle\sum_{m=i+1}^{j} \alpha_l^{Link}(i,m)\alpha_l^{Seq}(m,j) \end{aligned} \tag{3}$$

Outside The following probabilities measure the generation of word sequences $\mathrm{w}_{1,i-1}$ and $\mathrm{w}_{j+1,n}$ as words from i to j form a complete-sequence:

$$\begin{aligned}\beta_r^{Seq}(i,j) = &\sum_{h=j+1}^{n} \beta_r^{Seq}(i,h)\alpha_r^{Link}(j,h) \\ &+ \beta_r^{Link}(i,h)\alpha_l^{Seq}(j+1,h)p(w_i \rightarrow w_h) \\ &+ \beta_l^{Link}(i,h)\alpha_l^{Seq}(j+1,h)p(w_i \leftarrow w_h)\end{aligned} \tag{4}$$

$$\begin{aligned}\beta_l^{Seq}(i,j) = &\sum_{v=1}^{i-1} \beta_l^{Seq}(v,j)\alpha_l^{Link}(v,i) \\ &+ \beta_r^{Link}(v,j)\alpha_r^{Seq}(v,i-1)p(w_v \rightarrow w_j) \\ &+ \beta_l^{Link}(v,j)\alpha_r^{Seq}(v,i-1)p(w_v \leftarrow w_j)\end{aligned} \tag{5}$$

These probabilities calculate the formation of words preceding i and following j in a sentence when complete-link(i,j) results in $\mathrm{w}_{i,j}$:

$$\begin{aligned}\beta_r^{Link}(i,j) &= \textstyle\sum_{v=1}^{i} \beta_r^{Seq}(v,j)\alpha_r^{Seq}(v,i) \\ \beta_l^{Link}(i,j) &= \textstyle\sum_{h=j}^{n} \beta_l^{Seq}(i,h)\alpha_l^{Seq}(j,h)\end{aligned} \tag{6}$$

Probability Updates The iterative construction process for complete-links and complete-sequences continues until the dependency structure of the entire sentence, denoted as $Link_r(1,n)$, is built. Here, 1 and n respectively signify the positions of the first and last words, marking the beginning and end of the sentence. At this point, the dependency probabilities for w_i and w_j within the sentence can be determined as follows:

$$\begin{aligned}P(w_i \rightarrow w_j) &= \frac{\alpha_r^{Link}(i,j)\beta_r^{Link}(i,j)}{P(Link_r(1,n))} \\ P(w_i \leftarrow w_j) &= \frac{\alpha_l^{Link}(i,j)\beta_l^{Alnk}(i,j)}{P(Link_r(1,n))}\end{aligned} \tag{7}$$

When the calculations for each sentence in a corpus are complete, the resulting dependency probabilities between word pair (i,j) must be normalized as follows:

$$\begin{aligned}P^{'}(w_i \rightarrow w_j) &= \frac{P(w_i \rightarrow w_j)}{\sum_{k \in vocab} P(w_i \rightarrow w_k)} \\ P^{'}(w_i \leftarrow w_j) &= \frac{P(w_i \leftarrow w_j)}{\sum_{k \in vocab} P(w_i \leftarrow w_k)}\end{aligned} \tag{8}$$

Here, *vocab* represents the whole vocabulary of the corpus. This ensures that the total sum of all potential link probabilities from word i to every other word in the vocabulary, whether leftward or rightward, equals 1.

B Similarity Analysis of Selected Low- and High-Resource Languages

Due to the inherent similarities in vocabulary, grammar, and phonetics among languages from the same family, and the enhancement of performance across various tasks through shared cross-lingual information during multilingual model training (Kong et al. 2021; Üstün et al. 2022; Chronopoulou et al. 2023), this section will analyze the similarity between the eight low-resource languages selected in Section 4 and other languages for which training data are available.

To facilitate this analysis, we employ lang2vec (Littell et al. 2017), a powerful tool in natural language processing that offers standardized vectors. These vectors represent various language information types, providing detailed identifications for languages drawn from typological and phylogenetic databases. In our study, we utilize lang2vec to calculate the similarity between all languages involved in the training, and then visualize these similarities after dimensionality reduction with t-SNE (van der Maaten and Hinton 2008), as demonstrated in Figure 12. Visual analysis reveals that each selected low-resource language, which enables potential cross-lingual learning opportunities for these languages.

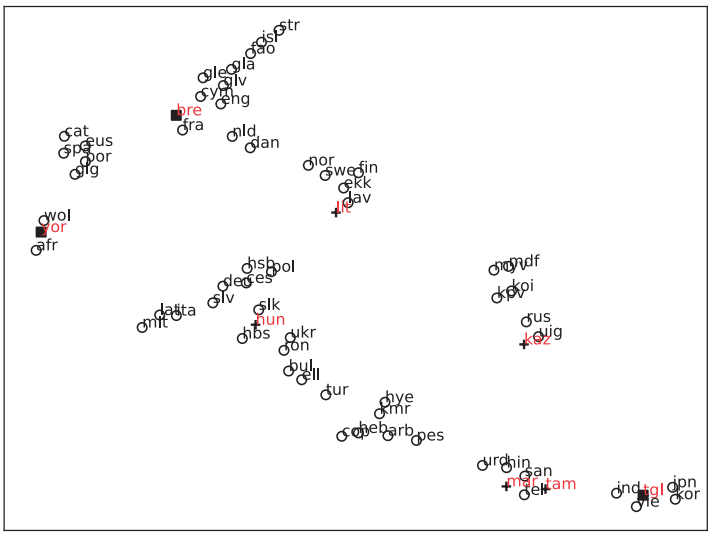

Figure 12 Visualize the language vectors for the languages used in UD v2.13 experiments. "■" represents the selected zero-shot languages, "+" represents the selected few-shot languages, and "○" represents other languages involved in the training. Selected low-resource languages are marked in red with abbreviations following the ISO 639-3. Note that there are 65 languages with training data in UD v2.13, but not every language is represented in the lang2vec language vectors, so the figure does not include all languages with training data.

Beyond linguistic features, the extent of similarities in the surface vocabulary between languages is crucial. UDify uses the mBERT tokenizer to process input sentences, thereby allowing sentences from various languages to share common tokens. To visually demonstrate the similarity in token usage among different languages, we use Breton, which employs the Latin alphabet, as an example. We calculated the token coverage rate between Breton and high-resource language treebanks by analyzing the proportion of shared tokens in their respective datasets. This method assessed their similarity, with results displayed in Figure 13.

The figure illustrates that Breton shares considerable similarity with other high-resource languages, which also use the Latin script. This similarity further supports the potential for cross-lingual learning. Additionally, the coverage rates of training sets from other high-resource language treebanks over their respective test sets range from 0.18 to 0.55, with an average of 0.31. Our constructed dataset D_{gen}^{Breton} achieves a coverage rate of 0.39 over Breton-KEB$_{test}$, effectively enhancing the parsing accuracy for Breton, as demonstrated in our experimental results.

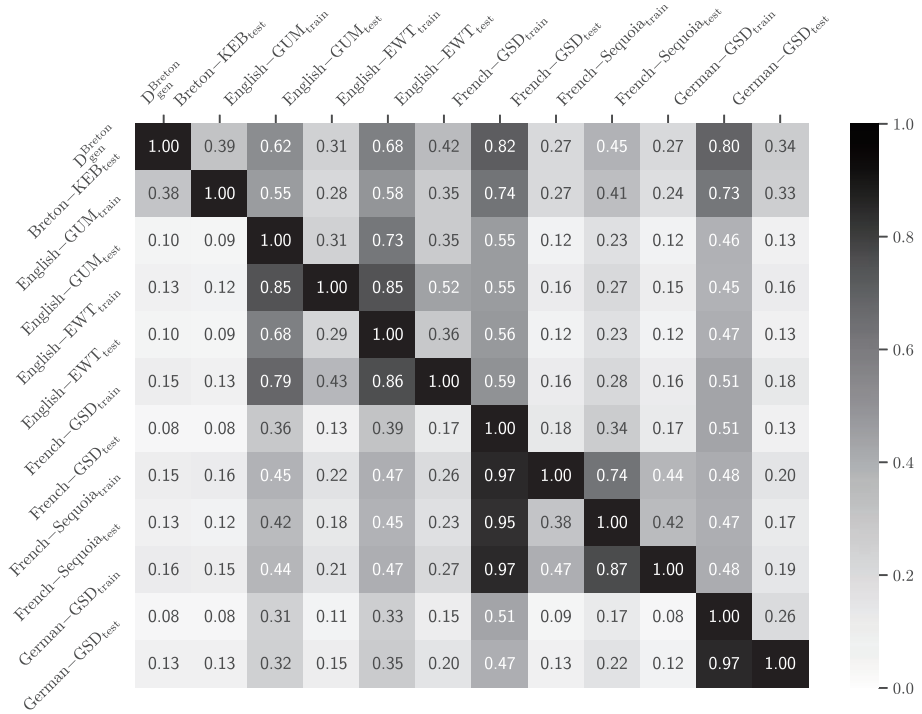

Figure 13 Bidirectional token coverage rates between train and test sets of different treebanks. Higher values indicate greater coverage and similarity between the datasets. The coverage is calculated using the formula $C(y, x) = \frac{|y \cap x|}{|y|}$, where $|y \cap x|$ represents the number of common tokens, and $|y|$ is the total number of tokens in the reference set.

C Experimental Results for LAS

We have placed the LAS results in Table 8, and conducted significance tests. In the UD v2.13 version for the eight languages involved in the experiment, 9 out of 11 treebanks showed improvement in LAS, with three zero-shot languages achieving significant enhancements.

For the `Tagalog-TRG` in the UD v2.3, the dataset was quite small, encompassing only 55 sentences. In the UD v2.13, the size of the test set was expanded to 128 sentences, resulting in significant differences in the improvement of the LAS between the two versions.

Experimental results indicate that Unsupervised Augmentation not only enhances the parsing accuracy of dependency arcs but also improves the parsing accuracy of dependency relations. However, because our proposed method focuses on generating dependency arc data through unsupervised learning, it does not specifically optimize or enhance the data for dependency relations. Consequently, despite the overall improvement in accuracy, there remains room for improvement in the fine-grained identification of dependency relation types. Future research could consider developing more detailed data augmentation techniques, particularly targeted enhancements for dependency relation types, to further improve the model's parsing accuracy.

	Code	v2.3				v2.13		
		UDify(org)	UDify(our)	Self	Unsup	UDify(our)	Self	Unsup
Few-shot	hu	84.9	**85.3**	85.3	85.1	86.0	**86.2**	86.1
	kk	63.7	65.1	64.9	**65.2**	64.3	63.9	**64.5**
	lt-a	—	—	—	—	76.7	**77.0**	76.9
	lt-h	**69.3**	69.2	68.9	68.3	**67.8**	66.6	67.5
	mr	**67.7**	62.9	62.4	63.6	60.0	61.7	**61.9**†
	ta-m	—	—	—	—	73.2	73.6	**73.7**
	ta-t	71.3	**71.4**	71.2	70.8	67.3	**67.5**	67.2
Zero-shot	br	39.8	50.1	50.5	**56.4**‡	47.3	47.0	**52.6**‡
	tl-t	40.1	57.2	52.4	**72.3**‡	44.3	46.3	**53.3**‡
	tl-u	—	—	—	—	32.5	34.0	**41.0**‡
	yo	19.1	22.0	22.0	**24.7**	23.2	23.5	**25.0**†
Few		—	**74.6**	74.5	74.5	74.8	74.9	**75.0**
Zero		—	44.5	44.7	**50.3**‡	36.5	36.6	**40.6**‡

Table 8 LAS (%) for few- and zero-shot languages obtained using different methods on UD v2.3 and v2.13. The UDify(org) result for the v2.3 was reported in Kondratyuk and Straka (2019). In the table, "‡" indicates a significance level of $p \leq 0.01$ and "†" indicates $p \leq 0.05$, both marks show instances where Unsup outperforms UDify(our).

Jiannan Mao: Jiannan Mao received his B.S. degree in Information Management and Systems from Gansu University of Political Science and Law, China, in 2019, and his M.S. degree in Intelligence Science and Engineering from Gifu University, Japan, in 2022. He is currently pursuing a Ph.D. at Gifu University. His research interests include natural language processing, with a focus on neural machine translation and dependency parsing.

Chenchen Ding: Chenchen Ding is a senior researcher with the Advanced Translation Technology Laboratory, the National Institute of Information and Communications Technology (NICT), Japan, from 2022. He is also an associate professor with the Multilingual Natural Language Processing Laboratory, a collaborative laboratory between NICT and Nara Institute of Science and Technology, from 2024. He received an M.E. and Ph.D. from the University of Tsukuba in 2012 and 2015, respectively.

Hour Kaing: Hour Kaing is a researcher at the Advanced Translation Technology Laboratory, National Institute of Information and Communications Technology (NICT), Japan, since 2022. He received his M.Sc. from the University of Grenoble 1, France, in 2014, and his Ph.D. from the NARA Institute of Science and Technology, Japan, in 2022.

Hideki Tanaka: Hideki Tanaka is a Research Executive Director of the Advanced Translation Technology Laboratory of the National Institute of Information and Communications Technology (NICT) in Japan. Before NICT, he was with the Science and Technology Research Laboratories of NHK and Advanced Telecommunications Research Institute International (ATR). His research interest lies in Natural Language Processing and Machine Learning. He received a BS., MS., and Ph.D. from Kyushu University in 1982, 1984, and 1995 respectively.

Masao Utiyama: Masao Utiyama is the director with Advanced Translation Technology Laboratory, the National Institute of Information and Communi cations Technology, Japan from 2023. He completed his doctoral dissertation with the University of Tsukuba, in 1997. His main research field is machine translation.

Tadahiro Matsumoto: Tadahiro Matsumoto is an associate professor of the faculty of engineering at Gifu University. He received his B.E., M.E. and

Ph.D. degrees in engineering from Gifu University in 1985, 1987 and 2008, respectively. His research interests include natural language processing and machine translation of sign languages.

(Received July 25, 2024)
(Revised November 1, 2024)
(Accepted December 4, 2024)