Improving Zero-Shot Dependency Parsing by Unsupervised Learning

Jiannan Mao[1,2] Chenchen Ding[2] Hour Kaing[2] Hideki Tanaka[2]
Masao Utiyama[2] Tadahiro Matsumoto[1]
[1]Gifu University, Gifu, Japan [2]ASTREC, UCRI, NICT, Japan
{mao, tad}@mat.info.gifu-u.ac.jp
{chenchen.ding, hour_kaing, hideki.tanaka, mutiyama}@nict.go.jp

Abstract

UDify [1] 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 was 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 [2, 3]. 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. Many research [4, 5] have demonstrated that the performance in part-of-speech (POS) or dependency parsing can be improved by pairing languages with similar. This multilingual approach reduces the cost of training multiple models for a language group [6].

UDify [1] is a multi-task self-attention network fine-tuned on multilingual BERT (mBERT) [7] pre-trained embeddings, capable of producing annotations for any treebank from Universal Dependencies treebanks [8]. UDify exhibits strong and consistent performance across a wide range of languages and tasks such as lemmatization, POS tags, and dependency parsing. In addition to UDify, there are several comparable state-of-the-art methods[5, 9]. A problem highlighted by several related studies is the substantial discrepancy in the performance of these methods in zero-shot scenarios, even with identical training strategies, datasets, models, and evaluation methods [9, 10].

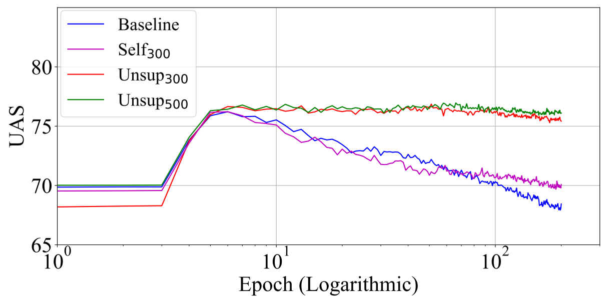

Figure 1 Change in the UAS of a model during the training process on the Breton test set for both the baselines (Baseline and Self) and the proposed method (Unsup).

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 are taken as a case study. Unsupervised learning-based data augmentation 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.

2 Background

2.1 UDify

The UDify model jointly predicts lemmas, POS tags, dependency structures, and etc. The pre-trained mBERT model is a self-attention network with 12 encoder layers. It is used in the UDify model for cross-lingual learning without additional tags to distinguish the languages. In addition, a strategy similar to that of ELMo [11] is adopted, where a weighted sum of the outputs of all layers is computed as follows and fed to a task-specific classifier:

$$e_j^{task} = \sum_i mBERT_{ij}.$$

Here, e^{task} denotes the contextual output embeddings for tasks such as the dependency parse, $mBERT_{ij}$ denotes the $mBERT$ representation for layer i at token position j.

In the task involving dependency structures, mBERT's subword tokenization process inputs words into multiple subword units. Despite this, only the embeddings e_j^{task} of the first subword unit are used, serving as input to the graph-based bi-affine attention classifier [2]. The resulting outputs are combined using bi-affine attention to produce a probability distribution of the arc-head for each word. Finally, the dependency tree is decoded using the Chu–Liu/Edmonds algorithm [12, 13].

2.2 Unsupervised Dependency Learning

Adhering to the properties of dependency syntax [14], a general unsupervised algorithm for projective N-gram dependency learning (Unsupervised-Dep) was described in Ding [15]. This method employs a CYK-style chart and dynamic programming to construct the optimal dependency tree, grounded in the non-constituent concepts of complete-link and complete-sequence. Due to the significant time complexity associated with N-gram learning, this study concentrates on bi-gram for practical applicability.

When considering the bi-gram, the directionality of a pair of words is set by the dependency relation, with $(w_i \rightarrow w_j)$ indicating a rightward relation and $(w_i \leftarrow w_j)$ indicating a leftward one. The bi-gram unsupervised learning update probabilities $P(w_i \rightarrow w_j)$ and $P(w_i \leftarrow w_j)$ are calculated using the Inside–Outside algorithm [16]. Finally, the Viterbi algorithm [17] is employed to determine the optimal tree construction.

3 Proposed Method

3.1 Motivation

In UDify's training, the dependency structures of zero-shot languages are learned through transfer learning. In contrast to the high-resource and few-shot languages, early saturation in the accuracy of the dependency parsing is exhibited during the learning process of all low-resource languages. The best performance was typically achieved around the 8th training epoch, as shown in Figure 2.

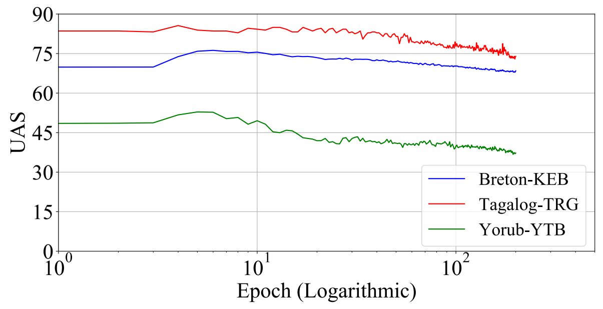

Figure 2 Changes in the UAS of zero-shot languages during the training process.

This phenomenon has been noticed [5, 9, 10] 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, we introduce a data augmentation technique grounded on Unsupervised-Dep, and aims to reduce the performance gap and thus improve the effectiveness of UDify in dependency parsing for zero-shot languages.

3.2 Unsupervised Augmentation

To apply Unsupervised-Dep in data augmentation, it is vital to ensure the generated data aligns with the UD format. Therefore, in addition to dependency arc-heads, other types of data must be created and combined with the results from Unsupervised-Dep. Given its high time complexity of $O(n^3)$, making 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 models [8, 3], providing a robust foundation for our initialization approach.

We initiate the process with the raw corpus, $Data$, input into the trained UDify by the original UD treebanks, to generate the dependency arc-heads, represented as DEP_{arc}, and POS, lemmas, etc., denoted as $Others$. Statistical computations on DEP_{arc} generate initial probabilities $P(w_i{\rightarrow}w_j)$ and $P(w_i{\leftarrow}w_j)$, serving as input for Unsupervised-Dep alongside $Data$.

Following several iterations of training through Unsupervised-Dep, the re-estimated $P(w_i{\rightarrow}w_j)'$ and $P(w_i{\leftarrow}w_j)'$ emerge. They become the parameters for the Viterbi algorithm to determine the optimal dependency arc-head as given by

$$DEP'_{arc} = Viterbi(x, P(w_i{\rightarrow}w_j)', P(w_i{\leftarrow}w_j)') \ ,$$

where DEP'_{arc} is the tree with the highest probability for a sentence x from $Data$.

We merged DEP'_{arc} with $Others$, ultimately generating artificial data. The artificial data are then combined with the existing UD treebanks for the subsequent training.

4 Experiments

4.1 Dataset

We selected Breton from OPUS [1] as our target zero-shot language for the implementation of Unsupervised-Dep. After cleaning the collected data, we obtained 99.5k sentences, which we refer to as OPUS-br.

From this collection, we first identified a subset of 300 sentences as our test set, 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 used for obtaining and updating the probabilities. The validation set is applied to validate the results of Unsupervised-Dep learning. The test set, demonstrates the results of our data augmentation.

1) opus.nlpl.eu

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

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

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 [18]. 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 [19], we follow the parameters and re-implemented the model provided by UDify[2], which we refer to as the Baseline. To expedite the experiment, we implemented multi-GPU parallel training[3] by modifying UDify using Horovod [20].

For our method, we initially computed the parsing results of the training data from OPUS-br using the Baseline. 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 unsupervised learning process of training data from OPUS-br. After the 10th training epoch, we employed the re-estimated probabilities to parse the test set from OPUS-br. In multilingual parser experiments, two different subsets of the OPUS-br dataset were employed to refine the dependency arc-heads generated by Unsupervised-Dep, specifically marked as Unsup$_{300}$ and Unsup$_{500}$. Unsup$_{300}$ integrates sentences from the OPUS-br test$_{300}$ into the UDify training set, while Unsup$_{500}$ utilizes sentences from the OPUS-br test$_{500}$.

Meanwhile, inspired by Rybak [21], we conducted an experiment using a comparative method dubbed Self$_{300}$. In this approach, we used the test$_{300}$, diverging from the

2) github.com/Hyperparticle/udify
3) When parallelizing across seven V100s, replicating UDify requires close to 12 days.

use of Unsupervised-Dep for sentence refinement. The sentences were directly parsed with the Baseline instead. The parsing results were then converted into UD format, merged with the original training set, and used for a new round of UDify model training.

4.3 Result and Discussion

A comparison with the experimental results reported in the original UDify study confirms that Baseline was re-implemented successfully, as demonstrated in Table 2. 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. Additionally, Figure 1 and Table 2 illustrates that, though different datasets have minimal impact on the peak UAS scores for Breton during training, methods Unsup$_{300}$ and Unsup$_{500}$ notably narrow the gap between the best and last UAS scores, thereby enhancing training stability.

Table 2 UD scores on Breton and other languages obtained by different methods. Rest(%) refers to the average score of UPOS, UFeats, Lemma, and LAS in the UD scores. The UDify result was reported by Kondratyuk [1].

	Breton				Other	
	UAS			Rest	UAS	Rest
	last	best	gap			
UDify	63.5	-	-	-	-	-
Baseline	68.4	76.2	-7.8	52.4	77.7	82.3
Self$_{300}$	70.1	76.2	-6.1	55.3	77.7	82.3
Unsup$_{300}$	75.4	76.5	-1.1	60.9	77.7	**82.4**
Unsup$_{500}$	**76.1**	**76.9**	**-0.8**	**61.1**	77.7	82.3

Considering UDify's role as a multilingual parser, it is necessary to evaluate the impact of the proposed method on other languages. To observe in detail the differences and changes in the UAS between the Baseline and Unsup$_{500}$, we conducted tests across all treebanks and display the results in Figure 3. From the figure, it is evident that while Unsup$_{500}$ has improved the UAS for Breton, it has had virtually no impact on the parsing precision of dependency constructions in other languages.

For a comprehensive comparison, the UD scores of the

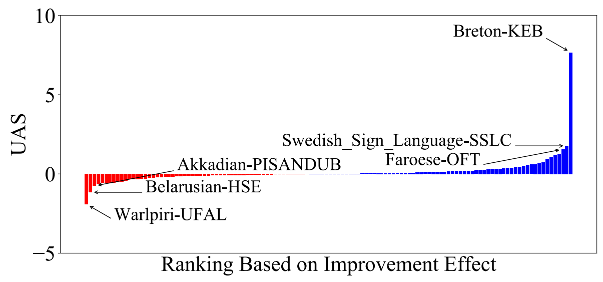

Figure 3 Changes in the UAS for the Baseline and Unsup$_{500}$ on all test treebanks. The x-axis sorts the UD treebanks by the ascending improvement of the proposed method over the Baseline.

Breton and other languages have been compiled in Table 2. Given that UDify must balance the loss produced by multiple decoders during training and Rybak's work [21], these variations in evaluation metrics are considered reasonable. Broadly, our data augmentation method has almost no negative impact on other languages and tasks, maintaining their performance levels.

Considering all results, we argue that generating training data for zero-shot languages through the application of the Unsupervised-Dep is both essential and effective in multilingual modeling.

5 Conclusion

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.

References

[1] Dan Kondratyuk and Milan Straka. 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, November 2019. Association for Computational Linguistics.

[2] Timothy Dozat and Christopher D. Manning. 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, 2017.

[3] Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 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, October 2018. Association for Computational Linguistics.

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

[5] Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. UDapter: Typology-based language adapters for multilingual dependency parsing and sequence labeling. **Computational Linguistics**, Vol. 48, No. 3, pp. 555–592, September 2022.

[6] 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. Google's multilingual neural machine translation system: Enabling zero-shot translation. **Transactions of the Association for Computational Linguistics**, Vol. 5, pp. 339–351, 2017.

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

[8] Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 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, October 2018. Association for Computational Linguistics.

[9] Anna Langedijk, Verna Dankers, Phillip Lippe, Sander Bos, Bryan Cardenas Guevara, Helen Yannakoudakis, and Ekaterina Shutova. 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)**, pp. 8503–8520, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[10] Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 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, November 2020. Association for Computational Linguistics.

[11] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)**, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[12] Yoeng-Jin Chu. On the shortest arborescence of a directed graph. **Scientia Sinica**, Vol. 14, pp. 1396–1400, 1965.

[13] Jack Edmonds, et al. Optimum branchings. **Journal of Research of the national Bureau of Standards B**, Vol. 71, No. 4, pp. 233–240, 1967.

[14] Jane J. Robinson. Dependency structures and transformational rules. **Language**, Vol. 46, No. 2, pp. 259–285, 1970.

[15] Chenchen Ding and Mikio Yamamoto. 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**, pp. 516–524, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

[16] Karim Lari and Steve J Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. **Computer speech & language**, Vol. 4, No. 1, pp. 35–56, 1990.

[17] G David Forney. The viterbi algorithm. **Proceedings of the IEEE**, Vol. 61, No. 3, pp. 268–278, 1973.

[18] Ryan McDonald, Slav Petrov, and Keith Hall. 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., July 2011. Association for Computational Linguistics.

[19] Martin Popel and Ondrej Bojar. Training tips for the transformer model. **Prague Bull. Math. Linguistics**, Vol. 110, pp. 43–70, 2018.

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

[21] Piotr Rybak and Alina Wróblewska. 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, October 2018. Association for Computational Linguistics.