Data Augmentation via Subtree Swapping for Dependency Parsing of Low-Resource Languages

Mathieu Dehouck **Carlos Gómez-Rodríguez**
Universidade da Coruña, CITIC
FASTPARSE Lab, LyS Research Group,
Departamento de Ciencias de la Computación y Tecnologías de la Información
Campus Elviña, s/n, 15071 A Coruña, Spain
{mathieu.dehouck,carlos.gomez}@udc.es

Abstract

The lack of annotated data is a big issue for building reliable NLP systems for most of the world's languages. But this problem can be alleviated by automatic data generation. In this paper, we present a new data augmentation method for artificially creating new dependency-annotated sentences. The main idea is to swap subtrees between annotated sentences while enforcing strong constraints on those trees to ensure maximal grammaticality of the new sentences. We also propose a method to perform low-resource experiments using resource-rich languages by mimicking low-resource languages by sampling sentences under a low-resource distribution. In a series of experiments, we show that our newly proposed data augmentation method outperforms previous proposals using the same basic inputs.

1 Introduction

Data sparsity has been a problem since the beginning of natural language processing. Neural networks have not solved it and have made it even more visible with their hunger for data. Hence a revived interest in data augmentation, since artificially created annotation can prove very useful to tackle the lack of manually annotated training data.

Many approaches have been proposed to perform data augmentation. Some rely on external resources, such as unannotated raw text in order to iteratively increase their training data with automatically annotated examples (McClosky et al., 2006; Yu and Bohnet, 2015). However, this is an error-prone method and useful annotations must be separated from harmful ones. Other proposals instead rely solely on available annotated data in order to generate new examples (Şahin and Steedman, 2018).

In this paper, we present a language-agnostic approach to the automatic generation of dependency-annotated sentences. Our method essentially swaps compatible subtrees from different sentences in order to generate new annotated sentences. By enforcing a number of constraints on the subtrees to be swapped, we avoid to generate too ungrammatical sentences. Furthermore, contrary to previous work that kept the syntactic structure of sentences or impoverished it, our method injects structures from other sentences, so it can introduce more syntactic complexity in the generated sentences.

In order to assess the potential of our new data augmentation method for low-resource languages in a way that is independent from the specific sentences in low-resource treebanks, we propose a method to mimic low-resource language data using high-resource language data. By sampling sentences from high-resource languages using a low-resource language distribution (over sentence length), we can perform the same experiment several times in a more faithful low-resource setting, while dampening the role played by each individual sentence on the final results, which can be prominent in very low-resource settings.

While there are many methods to parsing low-resource languages, such as unsupervised parsing or cross-lingual transfer, in this paper we consider a very restrained setting where one has only access to mono-lingual parsing data and nothing more. This very strict setting addresses two research problems.

This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: `http://creativecommons.org/licenses/by/4.0/`.

Proceedings of the 28th International Conference on Computational Linguistics, pages 3818–3830
Barcelona, Spain (Online), December 8-13, 2020

How to analyse a language that at the same time is an isolate and uses its own writing system (e.g. Japanese, Korean) at least in the data if we do not have typological information? And, how much can we learn from a very limited amount of training data?

2 Related Work

Automatically generated annotation has been used for dependency parsing for at least two decades. McClosky et al. (2006) used a combination of a k-best parser and a discriminative reranker in order to increase their training set size with automatically parsed sentences. Later McDonald et al. (2011) and Wróblewska and Przepiórkowski (2012) proposed various projection techniques to create annotated data for languages that did not have any, relying to different extents on parallel corpora.

Sentence morphing itself is not new either. Wang and Eisner (2018) and Rasooli and Collins (2019) proposed to reorder sentences from a source language in order to better match the word order of a target language. Aufrant et al. (2016) proposed not only to reorder words but also to delete some such as determiners when they are irrelevant for the target language.

However, these methods rely on external resources (parse trees in other languages and/or unparsed sentences from the same language) to create new data. And creating new examples this way introduces noise in the training data. In order to avoid this kind of problems, one can directly try to expand the little annotated data available.

Şahin and Steedman (2018) proposed to transform gold dependency trees directly by rotating their core arguments and deleting subtrees in order to generate new sentences for training POS taggers for low-resource languages. While similar in idea to their work, justifying experimental comparison, our work is different in two important aspects. First, their data generation methods and ours differ both in the grammaticality constraints and in the shape of the generated sentences. In addition, while the data generation process morphs dependency trees, they evaluated it on POS tagging. It was thus unclear how their data augmentation method would behave on dependency parsing, where the data generation process and the success measure both look at the same structure.

Later Vania et al. (2019) evaluated the data augmentation technique of Şahin and Steedman (2018) directly on dependency parsing as part of a wider investigation on methods for parsing low-resourced languages. Their results showed that creating sentences by morphing trees indeed helps parsers. They also considered replacing words by words with agreeing morphology to create new "nonce" sentences after Gulordava et al. (2018) and looked at cross-lingual training, which is also beneficial.

In this paper, we depart from their work by focusing on gold data morphing only. We propose a new data augmentation method that creates new "gold" parses from existing ones by swapping compatible subtrees. We also propose an experimental method for mimicking a low-resource setting using high resource languages.

3 Data Augmentation

In this section we present a language-agnostic method to automatically generate high-quality dependency annotated data from a small amount of (manually) annotated sentences. Beside being as language-agnostic as possible, our method should (1) create new structures and (2) create grammatically sound sentences.

While the cropping operation of Şahin and Steedman (2018) creates new sentences, it has two main issues. The first one is that sentences created this way may be ill-formed. For example, French finite verbs always need an overt subject, and removing it creates ungrammatical sentences. The second one is that it creates new sentences by impoverishing existing ones. Cropped trees are by essence smaller and simpler than their source trees. This may be a problem since small treebanks already have less material and simpler structures than bigger ones.

To avoid these issues, we propose to create new sentences by swapping subtrees between sentences under constraints we describe below. Swapping subtrees between sentences creates both simpler and more complex sentences, and strong constraints ensure that these sentences are as grammatical as possible.

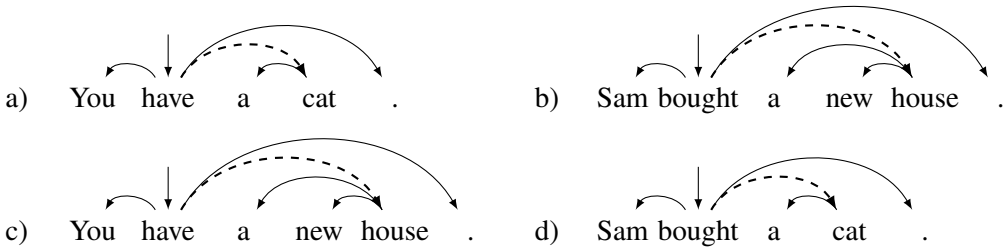

Figure 1: Illustration of tree swapping operation between sentences. Sentences *c* and *d* result from swapping the direct objects (represented with an incoming dashed arrow) from sentences *a* and *b*. Sentence *c* is more complex than *a* since its new direct object has an adjective.

Figure 1 shows an example of tree swapping that creates both a more complex and a simpler sentence. In sentence *c*, the new direct object has an adjective modifying the noun, while in *a*, the noun had just a determiner. Swapping subtrees can introduce other new complexities in otherwise simple sentences, such as relative clauses. Sentences *d* and *b* show the inverse phenomenon.

3.1 Grammatical Constraints

In the following, we assume Universal Dependencies annotations (UD) (Zeman et al., 2019), which we also use for the experiments, but the idea behind our data augmentation technique can be applied to other types of tree annotations given the relevant information is easily accessible to the head of a subtree.

To be as language-agnostic as possible while trying to keep maximal grammaticality, we enforce the root of the subtrees being swapped to have the same POS tag, morphological features and dependency relation. Of course, enforcing those three constraints at the same time is far too rigid for most languages, but we think it is a good compromise to provide a reasonable solution for every language, since knowing which constraint can be relaxed is highly language specific.

For example, as English or French do not mark case on their noun phrases, a subject noun phrase could be replaced by an object one as long as they agree in number (and sometimes in gender). However in Japanese, case is marked with clitics which in UD attach to their modifying nouns but the nouns themselves do not have any case marking. Therefore, swapping noun phrases whose roots agree in POS tag and morphological features, but not in dependency relation would often lead to ungrammatical sentences. Relaxing the morphological feature constraint would cause similar troubles in languages like Hebrew or Amharic in which verbs inflect for the gender and number of the subject. Finally, relaxing the POS tag constraint could cause troubles in Turkic languages where nominal direct objects can appear both in the nominative and accusative case while pronominal direct objects can only appear in the accusative.

3.2 Structural Constraints

In this paper, we only consider a subset of the available POS tags and dependency relations in order to guarantee the quality of the generated sentences. Regarding POS tags, we only consider `NOUN`, `PROPN`, `ADJ` and `VERB`. We avoid the other parts-of-speech because they could lead to more ungrammaticality. For example, replacing a form of the auxiliary (`AUX`) *to be* in a continuous construction with the corresponding form of *to have* would result in an ungrammatical sentence (*I am eating.* > **I have eating.*). Similarly, adverbs have such a wide range of uses that replacing any adverb by any other would likely yield incorrect sentences.

Regarding relations, we consider all core arguments (nominal and clausal), all non core dependents but `discourse`, `expl` and `dislocated`, and all nominal dependents. We ignore all the remaining relations, including conjunctions, multi-word expressions and so on, as defined in Universal Dependencies. This is detailed in the Appendix.

In order to make tree swapping easier, we also only consider projective subtrees and only allow one swap at a time. While we could generate more sentences by allowing multiple swaps, one swap already leads to a great amount of generated sentences so we only consider this case in this paper. Eventually, we

do not swap trees between a sentence and itself, to avoid intra-sentence redundancy. And we never swap main clause predicates (this is taken care for by avoiding the `root` relation) because it would not create new sentences, but rather duplicate them.

4 Low-resource Experiments with High-Resource Languages

To be able to perform extensive experiments, we chose to work with resource-rich languages and use them to mimic low-resource conditions. The most basic way to do so is to sample a small number of sentences from a big treebank to artificially create a small one. This has the big advantage that we can sample many different small treebanks and therefore dampen the influence of any given training sentence on parsing results. This is especially important in a very low-resource setting where each sentence can have a strong impact on the actual parsing results, while not being very representative of its actual source language.

However, the difference between high-resource treebanks and low-resource treebanks goes beyond the mere number of sentences or words. Bigger treebanks tend to have more varied constructions than their smaller counterparts[1]. This means that for a set number of sentences or tokens, a sample from a resource-rich treebank may contain more information than a sample from a resource-poor one. Therefore, if we want to mimic low-resource languages with high-resource ones, we have to take sentence complexity into consideration.

In this paper we use sentence length as a surrogate for sentence complexity, under the assumption that more complex constructions require more words[2]. Other complexity markers could be taken into account such as dependency relation distribution. But annotators do not choose sentences according to their internal structures or complexity, thus the relation distribution of a small treebank is as contingent as the presence of any given sentence in it. So we keep this open for future work.

4.1 Low-Resource Sampling

In order to mimic low-resource languages with high-resource ones, we sample data from high-resource languages using the macro averaged probability distribution of sentence length for languages that have less than 1000 training sentences overall. We truncate the distribution to sentences shorter than 100 tokens. Figure 2 represents the sentence length distribution of four sets of languages from UD 2.4. The dashed red line represents training sentences of languages with more than 1000 training sentences (over all the training sets when there are multiple treebanks). The dash dotted teal line represents training sentences of languages with less than 1000 training sentences (low-resource languages). In plain blue is a smoothed[3] version of the same distribution used to sample our training sets. In dotted orange is the distribution of test sentence length for languages that do not have a training set at all. While having a higher mode, the faster decay of the low-resource distribution gives it a lower mean than the resource-rich one. The test distribution of under-resourced languages (without a training set), is skewed toward short sentences, with both a low mean and a fast decay.

Language	En	Eu	Fi	He	Id	Ru	Ta *	Tr	Vi	Wo	Rich	Low	No Train
Treebank	EWT	BDT	FTB	HTB	GSD	Synt	TTB	IMST	VTB	WTB			
Train	16.3	13.5	8.5	26.3	21.8	17.8	15.8	10.3	14.5	19.8	18.0	17.0	-
Dev	12.6	13.4	8.4	23.6	22.6	18.0	15.8	10.2	14.4	22.9	18.6	18.6	-
Test	12.1	13.6	8.7	25.0	21.2	18.1	16.6	10.2	14.9	22.1	17.7	16.7	12.7

Table 1: Average sentence length of Train, Dev and Test sets of resource rich languages (Rich), low resource languages with less than 1000 training sentences (Low,*) and languages without training sentences (No Train).

[1]We believe that there are two main reasons behind this. One is the fact that shorter sentences are generally easier and faster to annotate. The other is genre distribution. Different genres have different informational needs and thus average sentence length. In UD 2.4, Tagalog and Warlpiri only have short grammatical examples.

[2]This is mostly true given a language. However, in a multilingual setting, typology plays an important role too, and a simple sentence in a more analytic language might require many more words than a more complex one in a more synthetic language. But as we use many different languages, we decide to use a one-size-fit-all solution to keep our analysis simpler.

[3]The distribution is smoothed using a discrete quadratic kernel and a window size of 5 as presented in (Rajagopalan and Lall, 1995).

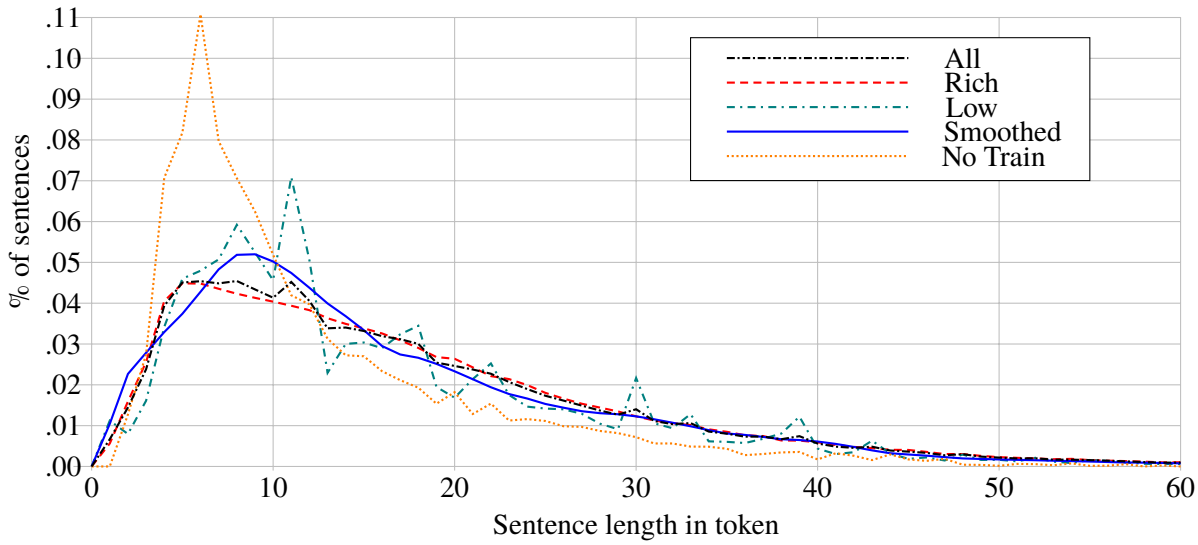

Figure 2: Macro average distribution of training sentence length for all languages, resource rich languages (more than 1000 training sentences), low-resource languages (less than 1000 training sentences), its smoothed version used for sampling sentences and test sentences of languages without a training set in UD 2.4.

Table 1 reports the average sentence length for the treebanks used in the experiments as well as for all resource-rich languages from UD 2.4 (more than 1000 training sentences), less-resourced languages (less than 1000 training sentences) and languages without training sentences. We see that average sentence length covers a range of values with short sentences in Finnish and 3 times longer in Hebrew. We also see that sentence length does not correlate with morphological complexity as on average Basque sentences are only one word shorter than Vietnamese ones, while Basque is morphologically much richer than Vietnamese. But from the three last lines, we see that as the quantity of resources decreases, so does the average sentence length. This is especially true for languages with no training data, where test sentences are 5 words (28.5%) shorter than those of resource-rich languages. This also means that test sets of better resourced languages must be more complex and more representative of their actual language. Thus, testing models trained with sub-sampled high-resource languages on their full test sets, should give a faithful lower bound of the scores achievable for really low-resource languages.

5 Experiments

In order to assess the potential of tree swapping as a data augmentation method for low-resource languages, we ran a series of experiments on 10 languages from UD 2.4, representing various families and various syntactical and morphological typologies. For each language, we sampled 8 times 40 sentences with the smoothed distribution of low-resource language sentence lengths depicted in Figure 2. While 40 training sentences might seem very low, in UD 2.4, Buryat has 19 training sentences, Kurmanji 20, Upper Sorbian 23 and Kazakh 31.

5.1 Quantity and Quality of Generated Sentences

We first start by looking at the number of new sentences generated by our method and compare it with the number of sentences generated by sentence cropping and rotation. We then look at the quality of the sentences generated by our method.

5.1.1 Number of Generated Sentences

While the constraints presented in Section 3 are very strict to prevent most non grammaticalities while staying language-agnostic, they still lead to a good number of new sentences. By definition, Şahin and Steedman (2018)'s crop and rotate produce linearly many new sentences with respect to the number of

Language	En	Eu	Fi	He	Id	Ru	Ta	Tr	Vi	Wo	Avg	
Treebank	EWT	BDT	FTB	HTB	GSD	Synt	TTB	IMST	VTB	WTB		×
Crop	46.4	57.3	33.9	57.9	57.6	59.0	65.4	50.6	52.9	56.5	53.7	1.3
Rotate	67.4	81.5	62.4	83.3	82.4	83.3	94.1	73.4	82.3	83.4	79.3	2.0
Swap	1604.8	990.6	611.5	870.6	1620.8	674.6	4794.0	1753.4	4563.6	2544.3	2002.8	50.0

Table 2: Average number of new sentences created by cropping, rotating and swapping dependency trees. For each language we sample 8 sets of 40 sentences, each following the same sentence length distribution, and use them as source to generate new sentences.

original sentences, but one swap creates quadratically many new sentences. If applied several times, tree swapping could generate arbitrarily many and arbitrarily big sentences, on the order of $\mathcal{O}(n^{k+1})$, where n is the number of original sentences and k the number of swaps. Table 2 reports the average number of sentences generated by Şahin and Steedman's crop and rotate operations and by our own tree swapping operation on datasets containing 40 sentences each.

As expected, we see that the number of new sentences generated by tree swapping is much higher than the number generated by crop and rotate. We also see that there is a correlation between morphological richness and the number of generated sentences. Morphologically rich languages (Basque, Finnish, Hebrew and Russian) have a rather low number of new sentences, less than one per pair ($n^2 = 1600$), meaning that not all sentence pairs have compatible subtrees because of the rigid constraints. On the other hand, morphologically simpler languages (such as Vietnamese) generate more sentences, almost 3 new sentences per pair, as there are less morphological constraints on subtrees.

Tamil is an outlier here, as it has the highest number of generated sentences despite being morphologically rich. Upon closer investigation, it turns out that UD Tamil data is relatively less complex than that of other morphologically rich languages. For example, in Russian sentence sets, there are 78 (POS, morphological features, dependency relation) triplets (with `NOUN` as POS) for 177 nouns (2.27 nouns per triplet), while in Tamil there are only 38 such triplets for 187 nouns (4.92 nouns per triplet). And Tamil data are much more skewed than those of other morphologically rich languages in terms of morphological feature distributions. The most frequent triplet where the POS is noun appears less than 6% of the time in Russian, but more than 23% of the time in Tamil.

5.1.2 Grammaticality of Generated Sentences

As we mentioned earlier, the strong constraints imposed on the trees to be swapped try to ensure that most generated sentences are grammatical. However, as it is hard to guarantee that every new sentence is actually valid, we looked at English and French sentences generated by the swap mechanism.

On a set of 100 sentences generated from the French Sequoia treebank, 14 could be considered problematic. Nine had an `nmod` noun phrase replaced by another one either lacking a determiner or inserting one in a context that does not allow it. Out of these nine sentences, three were actually due to dates in which the months were replaced by another noun phrase. Similarly, one sentence had a subordinated clause whose infinitive verb was missing an adposition. Three sentences were odd but not strictly ungrammatical due to adjectives that usually occur after their noun being placed before. Eventually, a sentence was containing a quote, and while the modification sounded odd inside a quote, outside it could be considered ungrammatical.

Out of 100 sentences generated from the English EWT treebank, 4 sentences were problematic. Two had problems with determiners after an adjective swap. In one case, the new adjective was starting with a consonant while the previous one started in a vowel, rendering the previous "*an*" irregular. In the other case, an adjective in a bare noun phrase was replaced by "*same*" which thus lacked a definite article. The other two were involving nested clauses. In one, a relative clause in which the governing noun filled the object role, was replaced by a relative clause with all its core arguments filled. In the other, a simple direct object was swapped with a direct object with a relative clause which in context was very odd.

Further exploration of the sentences generated for the experiments, revealed that adpositions are also a source of error, for example when swapping bare infinitives with *to*-infinitives. As this was already seen in French, we assume the same to be true for other languages. Examples of generated sentences are given

* He says , I have to have an Corporate ADDRESS .
* The forest phlox is blooming to Australia instead of mid-May .
Please send me an excel spreadsheet which depicts the Shrimp Scampi Dinner .
Do n't go , or you will learn how to waste any form of modern medicine .

Table 3: Examples of generated sentences. The newly inserted material is underlined. The second sentence is not strictly ungrammatical, but shows an example of adposition mismatch.

in Table 3. Problematic sentences are marked with an asterisk.

A reviewer mentioned their concerns about the fact that adpositions may be more important to grammaticality in other languages and that we should also enforce adpositions or other markers of the root to agree between sub-trees to be swapped. We agree with this remark and note that some UD treebanks start to be annotated with so called enhanced dependencies which, amongst other improvements, enhance relation labels with the lemma of the adposition that selects the phrase. Enhanced relations also encode information about the role of the governor in relative clauses. While the number of enhanced treebanks is still low, using enhanced relations instead of vanilla ones will further increase grammaticality of the created sentences.

This being said, the errors made by our method are actually interesting in that they question the notion of grammaticality. While a French or an English speaker would not produce them, if one was asked to analyse them within the UD framework, their analysis would likely be similar to the ones produced by our method.

We should also note that while 14 problematic sentences out of a hundred might seem a lot, it represents in fact 14 questionable arcs out of several hundreds. This opens a question that we would need to consider in future work. Namely, assuming that the test sentences are grammatical themselves, to which extent can the training data be ungrammatical.

5.2 Parsing Results

In a first experiment, we compare tree swapping with previous data augmentation techniques : tree cropping and rotation (Şahin and Steedman, 2018) and nonce sentences (Gulordava et al., 2018). As the number of new sentences that can be generated through tree swapping grows in $\mathcal{O}(n^{k+1})$, where n is the number of original sentences and k the number of swaps per sentence, we also look at the impact of the number of generated sentences. As the constraints for tree swapping are strong and therefore limit the number of sentences that can be generated, in a second experiment, we look at the impact of relaxing those constraints. Eventually, as the number of generated sentences gets big enough, it becomes possible to generate a separated development set for validating the trained models, so we also look at the impact of using an artificial development set.

We run each experiment 8 times (one for each sample). The reported results are LAS scores on the original dev sets averaged over those 8 runs. For all the experiments, we use an implementation of the biaffine parser of Dozat et al. (2017) available on Github[4]. We do not use pretrained word embeddings, so word and character representations are learned from scratch alongside the rest of the model. In this reduced data setting, we use 100 dimensions for word embeddings and 50 for characters. We only consider the task of parsing and use gold segmentation and UPOS tags.

5.2.1 Crop, Rotate, Nonce and Swap

Table 4 reports the average parsing results obtained on the development set using the base sampled training sets and augmented sets via cropping, rotating, nonce sentences and swapping. For this experiment, we use the maximum number of sentences generated by cropping and rotating. For nonce sentences and tree swapping, as it is easy to generate more sentences, we create n new sentences per original sentence. We create 4 nonce sentences per original sentence, for a total of 200 training sentences. For tree swapping, we experiment with $n \in \{2, 3, 4, 9\}$ effectively giving training sets of size 120, 160, 200 and 400. While

[4] https://github.com/zysite/biaffine-parser

Language		En	Eu	Fi	He	Id	Ru	Ta	Tr	Vi	Wo	Avg
Treebank		EWT	BDT	FTB	HTB	GSD	Synt	TTB	IMST	VTB	WTB	
Base	×1	28.24	22.38	16.83	37.95	27.12	22.09	27.05	14.68	14.79	26.65	23.78
Crop	×2.34	29.16	23.03	15.68	37.83	28.13	22.05	29.30	14.58	15.88	27.95	24.36
Rotate	×2.98	29.44	24.05	16.57	38.92	29.19	22.89	29.37	16.13	16.47	28.76	25.18
Nonce	×5	29.38	21.56	16.49	39.12	28.12	22.70	27.69	16.23	16.87	30.07	24.82
Swap	×3	31.67	24.36	16.69	38.19	30.11	23.28	29.35	16.34	17.55	30.33	25.79
	×4	31.47	23.52	16.46	39.21	29.94	**23.88**	**29.88**	16.68	18.34	31.26	26.06
	×5	**32.38**	24.76	17.21	**40.57**	30.52	23.00	29.42	16.57	18.33	31.77	26.45
	×10	31.06	**25.42**	**17.36**	39.85	**32.15**	23.76	29.69	**18.27**	**19.83**	**33.99**	**27.14**

Table 4: Average labeled attachment scores under various data augmentation techniques. Base is the results without data augmentation. Crop and Rotate are the corresponding operation from Şahin and Steedman (2018). Nonce uses sentences generated by replacing single words with compatible ones from other sentences as in Gulordava et al. (2018). Swap is our new subtree swapping method. Beside each method name is the average increase of number of sentences compared to the baseline.

this is more sentences than for crop and rotate, the original input is always the same 40 sentences, the difference comes from augmentation techniques and not from the data, so the comparison remains fair.

We see that on average, all augmentation techniques improve the results over the low-resource baseline. Swapping is consistently the best option, ignoring typological differences. More in detail, we have a strict ordering of crop, rotate and swap. Crop, which generates sentences with reduced complexity, has the lowest score of the three. Rotate, which keeps most of the original complexity but reorders it, is better. Swap, which potentially introduces new complexity, has the best score. As nonce is a weaker version of swap that only changes words in place and does not introduce new structures, it has a lower score than swap. This strongly suggests that a good data augmentation technique needs to create syntactic complexity.

Furthermore, while swap already beats rotate when they are allowed a comparable number of new sentences (2.98~3), swap can generate much more new sentences than rotate, and as we see, on average the score increases with the number of new sentences. This is interesting since not only does the new method generate a large number of sentences, but the models actually benefit from those sentences.

5.2.2 Constraints

Language	En	Eu	Fi	He	Id	Ru	Ta	Tr	Vi	Wo	Avg
Treebank	EWT	BDT	FTB	HTB	GSD	Synt	TTB	IMST	VTB	WTB	-
Base	28.24	22.38	16.83	37.95	27.12	22.09	27.05	14.68	14.79	26.65	23.78
PMR	**32.38**	24.76	17.21	40.57	30.52	23.00	29.42	16.57	18.33	31.77	26.45
PM	31.11	24.25	**17.62**	39.33	31.48	23.53	29.38	17.50	**19.18**	**32.70**	26.61
P R	31.34	24.01	16.11	**40.74**	30.61	23.90	29.61	16.83	18.33	32.13	26.36
MR	32.12	24.66	16.27	39.13	**31.85**	22.96	30.06	16.51	18.11	32.48	26.42
P	31.95	24.44	15.69	39.76	31.34	**24.61**	29.68	**17.70**	**19.18**	32.58	**26.69**
M	31.48	25.03	16.54	40.03	30.24	23.02	29.90	16.67	17.52	32.67	26.31
R	31.98	**25.48**	16.92	39.63	30.88	23.65	**30.50**	17.36	18.11	31.70	26.62
None	30.56	23.52	15.58	40.69	30.03	22.98	29.58	17.22	17.52	32.42	26.01
Best	32.38	25.48	17.62	40.74	31.85	24.61	30.50	17.70	19.18	32.7	27.28
Set	PMR	R	PM	P R	MR	P	R	P	PM	PM	-

Table 5: Average parsing results for different data augmentation constraints. P means that the heads of swapped trees must agree in POS tags. M means that they must agree in morphological features. R means that they must agree in dependency relation. None means that no such constraints apply. The last row is the best of each columns, assuming selection on a validation set. Each original sentence receives 4 new ones.

Table 5 reports parsing results obtained on the development set with augmented training sets via tree

swapping under all the possible constraint relaxations. Note that while POS tag and morphological features are tied to the head of a subtree, the dependency relation depends on the head of the subtree, but also on its new head and their relative position. We therefore choose to assign the relation of the original subtree's head to the newly swapped subtree's head. This might not be the best choice in all circumstances and is certainly very language specific.

We notice several interesting things. First, data augmentation is substantially beneficial for all but one language (Finnish) under all constraints. Then, over all the augmented settings, the one without constraints (last column) has the lowest score. This shows that constraints are important, supposedly because they help preserve grammaticality. And, as expected, not all constraints fit all languages equally well.

While some language/constraint scores might be surprising, it is important to note that we only partially relax those constraints. As we focus on a rather small subset of POS tags and on core dependency relations, relaxing some constraints might not have as strong an effect as if we had allowed the whole range of tree swapping in the first place. Even when relaxing the POS constraint, we only allow verbs, nouns, proper nouns and adjectives to move. Furthermore, in some languages there might be a strong overlap between relations and POS tags or relations and morphological features, dampening even more the effect of relaxing a constraint. For example, in a language where nouns inflect for case but verbs do not, morphological features alone will most of the time be sufficient to ensure that we do not swap verbs with nouns, so the POS constraint is superfluous. In fact, the best constraint on average is just agreeing POS tags, but it doesn't work for all languages such as Tamil and Finnish. Yet from the last row, it seems that the best option is still to pick the best constraint set for each language independently via validation on held-out data.

Looking more closely at languages individually, general patterns are harder to find. Quite surprisingly, we see that morphological features alone are not a good constraint. For Vietnamese, which is a morphologically very simple language, morphological features do not add anything. In fact, its PM and P are almost identical and give the same score. For Russian, cases can be triggered by different prepositions, meaning that cases need not align with dependency relations, and therefore relying solely on case for swapping trees may lead to erroneous trees. This is all the more possible given that we reassign subtrees' head dependency relations, meaning that we can have a prepositional phrase labeled as a direct object based on the morphology of its head alone. We guess similar reasons are behind the specific patterns of each language.

Another point to note is that the biaffine parser used for our experiments does not encode morphological information directly, but uses a character based word representation to indirectly represent morphological features. However, it has a dedicated POS tag embedding and uses dependency relation directly in its learning objective. That gives a different role to each constraint, with the morphological feature constraint being more remote from the parsing algorithm than the other two. We assume different parsing algorithms relying on different input representations would work differently with those constraints.

5.2.3 Importance of Development Data

When the number of sentences available is very low, it might not be possible to further split them into a training and a development set. This means that methods that need validations are more prone to overfitting as they will use the same data for training and validation. However, our data augmentation method creates enough new data so that it becomes possible to have a dedicated validation set distinct from the training set. But as this validation set is generated from the same original data as the training set, we need to see if it fulfills its role.

Table 6 reports results of parsers trained using different validation sets. The two middle rows (Train and Dev) are results using data augmented from the same original sentences. Train uses the same augmented set of 200 sentences for both training and validation (as does Base, but Base only has access the 40 original sentences), while Dev uses a distinct development set created for validation purpose. For the last row, the models were validated with the validation set created from another sentence sample.

Whether training benefits from a distinct validation set is language dependent and on average not significant. However, as expected, using a validation set based on other sentences is beneficial.

This is interesting since data augmentation is otherwise useful. The newly created sentences seem

Language	En	Eu	Fi	He	Id	Ru	Ta	Tr	Vi	Wo	Avg
Treebank	EWT	BDT	FTB	HTB	GSD	Synt	TTB	IMST	VTB	WTB	-
Base	28.24	22.38	16.83	37.95	27.12	22.09	27.05	14.68	14.79	26.65	23.78
Train	**32.38**	**24.76**	**17.21**	**40.57**	30.52	23.00	**29.42**	16.57	18.33	31.77	26.45
Dev	31.88	**24.76**	16.44	40.22	**30.92**	**23.17**	**29.42**	**16.66**	**19.08**	**32.02**	26.46
Other	**33.13**	**25.21**	**18.78**	**40.57**	**32.88**	**25.33**	**30.74**	**17.34**	**19.38**	**33.07**	27.64

Table 6: Average parsing results using various data sets as validation sets. Train uses the augmented training set for both training and validation of the model. Dev uses a distinct development set created from the same sentences as the train set for validation. Other uses a development set created from another set of 40 sentences for validation.

diverse enough to improve the quality of the learnt models, but not enough to make a real difference when it comes to validation.

A likely explanation is that while the parser sees new structures in the augmented data, the basic underlying information, especially the vocabulary, remains the same. Thus models validated on a created development set (different structure, but same vocabulary) may be pushed to rely more on vocabulary than on the actual dependency structure. On the other hand, models validated on a different set of sentences (different structure and different vocabulary) are pushed to pay more attention to structure since they need to handle unseen words.

6 Conclusion

In this paper, we have presented a new language-agnostic data augmentation technique based on dependency subtree swapping for creating new dependency trees. We also presented a method for performing more faithful *low-resource* experiments using high-resource languages by sampling training data under a distribution that favors shorter sentences to mimic the sentence length distribution of low-resource languages.

We have shown that our newly proposed tree swapping method consistently outperforms previously proposed augmentation techniques based on tree morphing. We have also shown that our method can create many new sentences and that they are useful for parser training as the score increases with the number of sentences. This is important since contrary to previous tree morphing techniques, the number of sentences created by tree swapping is potentially unbounded. Then, we have shown that relaxing the strong swapping constraints on a per language basis further improves the results. But that the language/constraints relation is not necessarily clear. Finally, we saw that despite being useful for training parsers, the created sentences are not diverse enough to be useful for model validation.

Previous works have demonstrated the possibility of training parsers with incomplete annotation (Lacroix et al., 2016). As a few generated sentences may be odd sounding or slightly ungrammatical, it would be interesting to see how parsers fare when trained with sound trees over ungrammatical sentences. We keep it as a future work. We also need to further investigate the three-way interaction between languages, augmentation techniques and parsing algorithms, as apparently not all augmentation techniques fare as well for all languages. Mixing data augmentation policies might also have a positive impact. More generally, it would be interesting to see how far a parser can go with only a handful of annotated sentences.

Acknowledgments

This work has received funding from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from MINECO (ANSWER-ASAP, TIN2017-85160-C2-1-R), from Xunta de Galicia (ED431C 2020/11), and from Centro de Investigación de Galicia 'CITIC', funded by Xunta de Galicia and the European Union (European Regional Development Fund - Galicia 2014-2020 Program), by grant ED431G 2019/01.

References

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge. In *COLING 2016, the 26th International Conference on Computational Linguistics*, pages 119–130, Osaka, Japan. The COLING 2016 Organizing Committee.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada, August. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana, June. Association for Computational Linguistics.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *HLT-NAACL*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 152–159, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Balaji Rajagopalan and Upmanu Lall. 1995. A kernel estimator for discrete distributions.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium, October-November. Association for Computational Linguistics.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong Kong, China, November. Association for Computational Linguistics.

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

Alina Wróblewska and Adam Przepiórkowski. 2012. Induction of dependency structures based on weighted projection. In Ngoc-Thanh Nguyen, Kiem Hoang, and Piotr J drzejowicz, editors, *Computational Collective Intelligence. Technologies and Applications*, pages 364–374, Berlin, Heidelberg. Springer Berlin Heidelberg.

Juntao Yu and Bernd Bohnet. 2015. Exploring confidence-based self-training for multilingual dependency parsing in an under-resourced language scenario. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 350–358, Uppsala, Sweden, August. Uppsala University, Uppsala, Sweden.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Noëmi Aepli, Željko Agić, Lars Ahrenberg, Gabrielė Aleksandravičiūtė, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Elvis de Souza,

Arantza Diaz de Ilarraza, Carly Dickerson, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Johannes Heinecke, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Takumi Ikeda, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Markus Juutinen, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Kamil Kopacewicz, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Maria Liovina, Yuan Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Tomohiko Morioka, Shinsuke Mori, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Daria Petrova, Slav Petrov, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Takaaki Tanaka, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Manying Zhang, and Hanzhi Zhu. 2019. Universal dependencies 2.5. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

A Appendices

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc compound	fixed flat	list parataxis reparandum	orphan goeswith dep	punct root

Table 7: Universal Dependency relations. The relations used to identify usable subtrees are on a white background. The excluded relations are on a gray background.