To Augment or Not to Augment? A Comparative Study on Text Augmentation Techniques for Low-Resource NLP

Gözde Gül Şahin
Koç University
Computer Science and
Engineering Department
gosahin@ku.edu.tr

Data-hungry deep neural networks have established themselves as the de facto standard for many NLP tasks, including the traditional sequence tagging ones. Despite their state-of-the-art performance on high-resource languages, they still fall behind their statistical counterparts in low-resource scenarios. One methodology to counterattack this problem is text augmentation, that is, generating new synthetic training data points from existing data. Although NLP has recently witnessed several new textual augmentation techniques, the field still lacks a systematic performance analysis on a diverse set of languages and sequence tagging tasks. To fill this gap, we investigate three categories of text augmentation methodologies that perform changes on the syntax (e.g., cropping sub-sentences), token (e.g., random word insertion), and character (e.g., character swapping) levels. We systematically compare the methods on part-of-speech tagging, dependency parsing, and semantic role labeling for a diverse set of language families using various models, including the architectures that rely on pretrained multilingual contextualized language models such as `mBERT`*. Augmentation most significantly improves dependency parsing, followed by part-of-speech tagging and semantic role labeling. We find the experimented techniques to be effective on morphologically rich languages in general rather than analytic languages such as Vietnamese. Our results suggest that the augmentation techniques can further improve over strong baselines based on* `mBERT`*, especially for dependency parsing. We identify the character-level methods as the most consistent performers, while synonym replacement and syntactic augmenters provide inconsistent improvements. Finally, we discuss that the results most heavily depend on the task, language pair (e.g., syntactic-level techniques mostly benefit higher-level tasks and morphologically richer languages), and model type (e.g., token-level augmentation provides significant improvements for* `BPE`*, while character-level ones give generally higher scores for* `char` *and* `mBERT` *based models).*

Submission received: 15 January 2021; revised version received: 19 August 2021; accepted for publication: 7 October 2021.

https://doi.org/10.1162/COLI_a_00425

1. Introduction

Recent advancements in the natural language processing (NLP) field have led to models that surpass all previous results on a range of high-level downstream applications, such as machine translation, text classification, dependency parsing, and many more. However, these models require a huge number of training data points to achieve state-of-the-art scores and are known to suffer from the out-of-domain problem. In other words, they are not able to correctly label or generate novel, unseen data points. In order to boost the performance of such systems in the presence of low data, the researchers have introduced various *data augmentation* techniques that aim to increase the sample size and also the variation of the lexical (Wei and Zou 2019; Fadaee, Bisazza, and Monz 2017; Kobayashi 2018; Karpukhin et al. 2019) or syntactic patterns (Vickrey and Koller 2008; Şahin and Steedman 2018; Gulordava et al. 2018). A similar line of research introduced adversarial attack and defense mechanisms (Belinkov and Bisk 2018; Karpukhin et al. 2019) based on injecting noises with the goal of more robust NLP systems.

Even though low-resource languages are the perfect test bed for such augmentation techniques, a large number of studies only *simulate* a low-resource environment by sampling from a high-resource language like English (Wei and Zou 2019; Guo, Mao, and Zhang 2019a). Unfortunately, methods that perform well on English may function poorly for many low-resource languages. This is due to English being an analytic language while most low-resource languages are synthetic. Furthermore, the majority of the studies focus either on sentence classification or machine translation. Although these tasks are important, the traditional sequence tagging tasks where tokens (e.g., white-`Adj` cat-`Noun`) or the relation between tokens (e.g., white `Modifier` cat) are labeled are still considered as primary steps to natural language understanding. These tasks generally have finer-grained labels and are more sensitive to noise. In addition, previous work mostly report single best scores that can be achieved by the augmentation techniques.

To extend the knowledge of the NLP community on text augmentation methods, there is a need for a comprehensive study that investigates (i) different types of augmentation methods on (ii) a diverse set of low-resource languages, for (iii) a diverse set of sequence tagging tasks that require different linguistic skills (e.g., morphologic, syntactic, semantic). In order to address these issues, we explore a wide range of text augmentation methodologies that augment on the character level (Karpukhin et al. 2019), token level (Kolomiyets, Bethard, and Moens 2011; Zhang, Zhao, and LeCun 2015; Wang and Yang 2015; Wei and Zou 2019) and syntactic level (Şahin and Steedman 2018; Gulordava et al. 2018). To gain insights on the capability of these methods, we experiment on sequence tagging tasks of varying difficulty: POS tagging and dependency parsing and semantic role labeling, also known as shallow semantic parsing. POS tagging and dependency parsing experiments are performed on truly low-resource languages that are members of various language families: Kazakh (Turkic), Tamil (Dravidian), Buryat (Mongolic), Telugu (Dravidian), Vietnamese (Austro-Asiatic), Kurmanji (Iranian), and Belarusian (Slavic). Due to the lack of annotated data, we simulate the low-resource environment for semantic role labeling (SRL) on languages from a diverse set of families, namely, Turkish (Turkic), Finnish (Uralic), Catalan (Romance), Spanish (Romance), and Czech (Slavic). To investigate whether the augmentations are model-agnostic and can further improve on state-of-the-art models, we experiment with distinct models that use various subword units (e.g., character, byte-pair-encoding, and word piece), pretrained embeddings (e.g., BPE-GloVe, multilingual BERT), and architectural designs (e.g., biaffine, transition-based parser). We train and test each model multiple times and report the mean and standard deviation scores along with the p-values calculated via paired t-tests. Furthermore, we investigate the sensitivity of augmentation techniques to parameters in a separate study, and analyze the contribution of each technique to the improvement of frequent and rare tokens and token classes.

Our results show that augmentation methods benefit the dependency parsing task more significantly than part-of-speech tagging and semantic role labeling—in that given order—independent from the parser type. The improvements are more reliably observed in morphologically richer languages; that is, the results for Vietnamese (an analytic language) are varied—in some cases significantly worse than the baselines. We find that augmentation can still provide performance gains over the strong baselines built on top of pretrained contextualized language models—especially in dependency parsing. In general, character-level augmentation gives more consistent improvements for the experimented languages and tasks, whereas synonym replacement and rotating sentences mostly result in a weak increase or decrease. We find that the subword unit choice and task requirements also have a large impact on the results. For instance, we observe significant gains over the character-based POS tagger, but no improvement for the BPE-based one for Tamil language. In addition, we find that token-level augmentation is more effective for BPE-based models, whereas character-level augmentation techniques provide significant gains for the rest. Furthermore, we observe that syntactic augmentation techniques are more likely to improve the performance on morphologically richer languages as pronounced in the SRL task.

2. Related Work

Augmentation techniques for NLP tasks fall into two categories: feature space augmentation (FSA) and text augmentation (TA). FSA techniques mostly focus on augmenting the continuous representation space directly inside the model, while TA processes the discrete variables such as raw or annotated text.

FSA. Guo, Mao, and Zhang (2019b) and Guo (2020) propose to generate a synthetic sample in the feature space via linearly and nonlinearly interpolating the original training samples. Although the idea originates from the computer vision field (Zhang et al. 2018), it has been adapted to English sentence classification tasks successfully (Guo, Mao, and Zhang 2019a). More recently Guo, Kim, and Rush (2020) proposed a similar technique that mixes up the input and the output sequences and showed improvements on several sequence-to-sequence tasks such as machine translation between high-resource language pairs. Despite their success in text classification and sequence-to-sequence tasks, they are seldom used for sequence tagging tasks. Zhang, Yu, and Zhang (2020) use the mixup technique (Zhang et al. 2018) in the scope of active learning, where they augment the queries at each iteration and later classify whether the augmented query is plausible—since the resulting queries might be noisy—and report improvements for the Named Entity Recognition (NER) and event detection task. However, building a robust discriminator for more challenging tasks as dependency parsing (DP) and SRL is a challenge on its own. Chen et al. (2020a) report that direct application of the mixup techniques (Guo, Mao, and Zhang 2019b; Guo 2020) for NER introduces vast amounts of noise, therefore making the learning process even more challenging. To address this, Chen et al. (2020a) introduce local interpolation strategies that mixes sequences *close* to each other, where *closeness* for tokens is defined either as (i) occurring in the same sentence (ii) occurring in the sentence within the k_{th} neighborhood, and show improvements over state-of-the-art NER models. Even though NER is a sequence tagging task, it is fundamentally different from the tasks in this study: DP and SRL. The local context for the NER task is usually the full sentence; however, for SRL the local context is much smaller. Imagine these sentences: "I want to visit the *USA*" and "*USA* is trying to solve the problems". NER would label *USA* as COUNTRY in both sentences, whereas *USA* would have the labels ARG1: VISITED and ARG0: AGENT/ENTITY TRYING. As can be seen, for our sequence tagging tasks, the local context is much smaller and the labels are more fine-grained than in NER. Therefore, Chen et al. (2020a) would have an extremely small sam ple space, whereas Chen et al. (2020b) is likely to produce too much noise. To sum up, using these techniques for more complicated sequence tagging tasks such as DP and SRL is not straightforward. This is due to (i) *the relations between tokens* being labeled instead of the tokens themselves; and (ii) the labels being extremely fine-grained compared to classification tasks. Furthermore, FSA techniques require direct access to the neural architecture since they modify the embedding space. That means they cannot be used in combination with black-box systems, which might be a drawback for NLP practitioners. Finally, unlike the aforementioned sequence labeling tasks, DP and SRL have a relatively complex input layer where additional linguistic information (e.g., postags, predicate flag) is used. That also makes the direct application of FSA techniques more challenging. For these reasons, we focus on the other category of augmentation methods.

TA. Vickrey and Koller (2008) introduce a number of expert-designed sentence simplification rules to augment the data set with simplified sentences, and show improvements on English semantic role labeling. Similarly, Şahin and Steedman (2018) propose an automated approach to generated simplified and reordered sentences using dependency trees. We refer to such methods as *label-preserving* because they do not alter the semantics of the original sentence. They mostly perform on syntactically annotated data and sometimes require manually designed rules (Vickrey and Koller 2008).

Another set of techniques (Gulordava et al. 2018; Fadaee, Bisazza, and Monz 2017) performs lexical changes instead of syntactic restructuring. Gulordava et al. (2018) generate synthetic sentences by replacing a randomly chosen token with its *syntactic* equivalent. Fadaee, Bisazza, and Monz (2017) replace more frequent words with the rare ones to create stronger lexical-label associations. Wei and Zou (2019) and Zhang, Zhao, and LeCun (2015) make use of semantic lexicons like WordNet to replace words with their synonyms. Kobayashi (2018), Wu et al. (2019), and Fadaee, Bisazza, and Monz (2017) use pretrained language models to generate a set of candidates for a token position, while Anaby-Tavor et al. (2020), Kumar, Choudhary, and Cho (2020), and Ding et al. (2020) take a generative approach. Most work focuses on classification and translation, except for Ding et al. (2020), who experiment on low-resource tagging tasks. However they assume one label per token, therefore this is not directly applicable to relational tagging tasks such as dependency parsing and semantic role labeling. Additionally, Wei and Zou (2019) propose simple techniques such as adding, removing, or replacing random words and show that such perturbations improve the performance of English sentence classification tasks. Unlike other lexical augmentation methods, these do not require syntactic annotations, semantic lexicons, and large language models that make them more suitable to low-resource settings.

There also exist sentence-level techniques such as back-translation (Sennrich, Haddow, and Birch 2016a) that aim to generate a paraphrase by translating back and forth between a pair of languages. However, such techniques are mostly not applicable to sequential tagging tasks like semantic parsing while they cannot guarantee the token-label association. Furthermore, training an intermediate translation model for paraphrasing purposes may not be possible for genuinely low-resource languages. In a similar line, Yoo et al. (2021) leverage a pretrained large language model to generate text samples and report improvements on English text classification tasks.

Another category of TA uses *noising* techniques that are closely associated with adversarial attacks to text systems. Belinkov and Bisk (2018) attack machine translation systems by modifying the input with synthetic (e.g., swapping characters) and natural noise (e.g., injecting common spelling mistakes). Similarly Karpukhin et al. (2019) defend the machine translation system by training on a set of character-level synthetic noise models. Han et al. (2020) introduce an adversarial attack strategy for structured prediction tasks including dependency parsing. They first train a seq2seq generator via reinforcement learning where they design a special reward function that evaluates whether the generated sequence could trigger a wrong output. The evaluation is carried out by two other reference parsers—that is, if both parsers are *tricked* then it is likely that the victim parser would also be misled. Han et al. (2020) then use adversarial training as a defense mechanism and show improvements. In the same line of research, Zheng et al. (2020) propose replacing a token with a "similar" token, where similarity is defined as having a similar log likelihood of being generated by pretrained BERT (Devlin et al. 2019) given the context and having the same POS tag (in the black-box setting). The tokens are chosen in a way that the parser's error rate is maximized. The attack is called a "sentence-level attack" when the chosen tokens' positions are irrelevant. On a "phrase-level attack," the authors first choose two subtrees and then maximize the error rate on the target subtree by modifying the tokens in the source subtree. Even though the adversarial example generation techniques (Zheng et al. 2020; Han et al. 2020) could be used to augment data in theory, the requirements—such as a separate seq2seq generator, a BERT-based scorer (Zhang et al. 2020), reference parsers that are of certain quality, external POS taggers, and high quality pretrained BERT (Devlin et al. 2019) models—make them challenging to apply on low-resource languages. Besides, most of the aforementioned adversarial attacks are optimized to trigger an undesired change in the output with minimal modifications, while data augmentation is only concerned about increasing the generalization capacity of the model.

Surveys. Due to the growing number of data augmentation techniques and interest in them, a couple of survey studies have been recently published (Hedderich et al. 2021; Feng et al. 2021; Chen et al. 2021). Hedderich et al. (2021) give a comprehensive overview of recent methods that are proposed to tackle low-resource scenarios. The authors overview a range of techniques both for low-resource languages and domains, discussing their limitations, requirements, and outcomes. Apart from data augmentation methods, the authors discuss more general approaches such as cross-lingual projection, transfer learning, pretraining (of large multilingual models) and meta-learning, which are not in scope of this work. Feng et al. (2021) provide a more focused survey, zooming in on data augmentation techniques rather than other common approaches such as transfer learning. They categorize data augmentations into three categories as (i) rule-based, (ii) example interpolation, and (iii) model-based techniques. **Rule-based techniques** are referred to as easy-to-compute methods such as EDA (Wei and Zou 2019) and dependency tree morphing (Şahin and Steedman 2018), which are also covered in this study. **Example interpolation** techniques are used to define the FSA type of methods that we have discussed earlier in this section. **Model-based techniques** refer to augmentation techniques that rely on large models trained on large texts (e.g., backtranslation and synonym replacement using BERT-like models) that either don't preserve the labels or require a strong pretrained model—which is mostly not available for low-resource languages. Both Feng et al. (2021) and Hedderich et al.

(2021) provide a taxonomy and a structured bird's eye view on the topic without any empirical investigation as in this study. The closest work to ours is by Vania et al. (2019), who explore two of the augmentation techniques along with other approaches (e.g., transfer learning) for low-resource dependency parsing, which is limited in numbers of tasks, languages, and augmentation techniques. Finally, Chen et al. (2021) provide an empirical survey covering a wide range of NLP tasks and augmentation approaches where most of them exist for English only. Hence, the experiments are performed *only on English*, providing no insights for low-resource languages or sequence tagging tasks. Unlike previous literature, our work (i) focuses on sequential tagging tasks that require various linguistic skills: part-of-speech tagging, dependency parsing, and semantic role labeling; (ii) experiments on multiple low-resource languages from a diverse set of language families; and (iii) compares and analyzes a rich compilation of augmentation techniques that are suitable for low-resource languages and the focused tasks.

3. Augmentation Techniques

As discussed in Section 2, we categorize the augmentation techniques as *textual* (TA) and *feature-space* augmentation (FSA). In this article, we focus on *textual augmentation* techniques rather than FSA for several reasons. First of all, existing FSA techniques have only been implemented and tested for simple sequence tagging tasks where the task is to choose the best tag for the token from a quite limited number of labels. However, in our sequence tagging tasks, the labels are quite *fine-grained*, and the *relation between tokens* are labeled rather than the tokens themselves. Second, FSA techniques require direct access to the neural architecture because they modify the embedding space, which means that they cannot be used in combination with black-box systems. Finally, for some of the sequence tagging tasks, namely, as dependency parsing and semantic role labeling, the models might have relatively complex input layers (e.g., incorporate additional features such as postags or predicate flags), which makes the direct application of FSA techniques more challenging.

Textual augmentation techniques, that is, augmentation techniques that alter the input text, can be applied on many different levels such as character, token, sentence, or document. Because we focus on sentence-level tagging tasks, document-level augmentation techniques are simply ignored in this study. Furthermore, the focus of this article is to investigate techniques that are *suitable for low-resource languages* and are able to *preserve the task labels* up to some degree. For instance, sentence-level augmentation techniques like backtranslation are not suitable since they would not preserve the token labels. Similarly, genuinely low-resource languages do not have associated strong pretrained language models due to lack of raw data. Therefore sophisticated techniques that make use of such models are also left out in this article.

To provide more details, the overview of the DA techniques investigated in Feng et al. (2021) is given in Table 1 to justify our selection of techniques. Here, the first category refers to the techniques that are included in this study. They modify the data on the *input* level, are task agnostic, can preserve the token and relation labels—hence suitable for our sequence tagging tasks—and do not require large amounts of text or models. The methods in the second category (Sennrich, Haddow, and Birch 2016a; Vaibhav et al. 2019; Nguyen et al. 2020; Wieting and Gimpel 2017; Feng, Li, and Hoey 2019; Singh et al. 2019; Anaby-Tavor et al. 2020) are not able to preserve the labels. For instance, Semantic Text Exchange (STE) (Feng, Li, and Hoey 2019) aims to replace an entity in a given sentence while modifying the rest of the sentence accordingly. Given the input *"great food , large portions ! my family and i really enjoyed our saturday morning*

Table 1
Comparison of selected DA methods adapted from Feng et al. (2021). *Level* denotes the depth at which data is modified by the DA. *Task* refers to whether the DA method can be applied to different tasks (i.e., task-agnostic), or specifically designed for a task. The *Reason* column provides the reason why the method is not included in this article.

DA Method	Level	Task	Reason
SYNONYM REPLACEMENT (Wang and Yang 2015)	Input	Agnostic	included in study
RANDOM DELETION (Wei and Zou 2019)	Input	Agnostic	included in study
RANDOM SWAP (Wei and Zou 2019)	Input	Agnostic	included in study
DTREEMORPH (Şahin and Steedman 2018)	Input	Agnostic	included in study
SYNTHETIC NOISE (Karpukhin et al. 2019)	Input	Agnostic	included in study
NONCE (Gulordava et al. 2018)	Input	Agnostic	included in study
BACKTRANSLATION (Sennrich, Haddow, and Birch 2016a)	Input	Agnostic	labels not preserved
UBT & TBT (Vaibhav et al. 2019)	Input	Agnostic	labels not preserved
DATA DIVERSIFICATION (Nguyen et al. 2020)	Input	Agnostic	labels not preserved
SCPN (Wieting and Gimpel 2017)	Input	Agnostic	labels not preserved
SEMANTIC TEXT EXCHANGE (Feng, Li, and Hoey 2019)	Input	Agnostic	labels not preserved
XLDA (Singh et al. 2019)	Input	Agnostic	labels not preserved
LAMBADA (Anaby-Tavor et al. 2020)	Input	classification	labels not preserved
CONTEXTUALAUG (Kobayashi 2018)	Input	Agnostic	requires strong pretrained model
SOFT CONTEXTUAL DA (Gao et al. 2019)	Emb/Hidden	Agnostic	requires strong pretrained model
SLOT-SUB-LM (Louvan and Magnini 2020)	Input	slot filling	requires strong pretrained model
WN-HYPERS (Feng et al. 2020)	Input	Agnostic	requires WordNet
UEDIN-MS (DA part) (Grundkiewicz, Junczys-Dowmunt, and Heafield 2019)	Input	Agnostic	requires spell checker
SEQMIXUP (Guo, Kim, and Rush 2020)	Input	seq2seq	not suitable
EMIX (Jindal et al. 2020a)	Emb/Hidden	classification	not suitable
SPEECHMIX (Jindal et al. 2020b)	Emb/Hidden	Speech/Audio	not suitable
MIXTEXT (Chen, Yang, and Yang 2020b)	Emb/Hidden	classification	not suitable
SWITCHOUT (Wang et al. 2018)	Input	machine translation	not suitable
SIGNEDGRAPH (Chen, Ji, and Evans 2020)	Input	paraphrase	not suitable
DAGA (Ding et al. 2020)	Input+Label	sequence tagging	not suitable
SEQMIX (Zhang, Yu, and Zhang 2020)	Input+Label	active sequence labeling	not suitable
GECA (Andreas 2020)	Input	Agnostic	not suitable

breakfast" and the entity to be replaced as *pizza*, STE generates a new sentence *"80% great pizza, chewy crust ! nice ambiance and i really enjoyed it ."*. Because the generated sentence is both syntactically and semantically different from the original sentence, such techniques cannot be used for any of our tasks. Furthermore most of these techniques are tested on English language only and require large amounts of data to generate meaningful paraphrases. The next category of techniques (Kobayashi 2018; Gao et al. 2019; Louvan and Magnini 2020) benefit from large pretrained language models to replace a token/phrase. Even though such models might exist for some of the low-resource languages, the quality of the models are low due to insufficient training data. Therefore using these models introduces more noise than expected. The next category contains techniques (Feng et al. 2020; Grundkiewicz, Junczys-Dowmunt, and Heafield 2019) that require external tools/lexicons such as WordNet and spell checkers that are only available for high resource languages. The final category consists of the techniques that are tuned for a specific NLP task. Therefore they can only be used for the specific task, and not for the tasks we focus on in this study. One exception is GECA (Andreas 2020), which can be used for parsing low-resource languages. One issue with GECA is that the boundary of extracted fragments can exceed the constituency span, hence there is no guarantee that the fragment would be a subtree.

This section includes a detailed discussion on three main categories of textual augmentation techniques that are used in our experiments, namely, *syntactic*,

Table 2
Comparison of augmentation methods by means of their task viability. x: Can be used for the task; o: Cannot be used for the task. Augmentations are generated on: "I wrote him a letter".

		POS	DEP	SRL	Generated
Char	**Char Insert (CI)**	x	x	x	I wrrotle him a legtter
	Char Substitute (CSU)	x	x	x	I wyote him a lettep
	Char Swap (CSW)	x	x	x	I wtore him a lteter
	Char Delete (CD)	x	x	x	I wote him a leter
Token	**Synonym Replacement (SR)**	x	x	x	I wrote him a message
	RW Delete (RWD)	x	o	o	I him a letter
	RW Swap (RWS)	x	o	o	I him wrote a letter
	RW Insert (RWI)	o	o	o	I wrote him a her letter
Syntactic	**Crop**	x	x	x	I wrote a letter
	Rotate	x	x	x	Him a letter I wrote
	Nonce	x	x	o	I wrote him a flower

token level, and *character level*. A summary of the techniques under each category and their suitability to our downstream tasks can be found in Table 2. Finally, it discusses the parameters associated with each technique in Section 3.4.

3.1 Character-Level Augmentation

The idea of adding synthetic noise to text applications is not new; however, it has mostly been used for adversarial attacks or to develop more robust models (Belinkov and Bisk 2018; Karpukhin et al. 2019). Previous work by Karpukhin et al. (2019) introduces four types of synthetic noise on orthographic level: character deletion (CD), insertion (CI), substitution (CSU), and swapping (CSW). Additionally, they introduce a mixture of all noise types by sampling from a distribution of 60% clean (no noise) and 10% from each type of noise, which we refer to as *Character All* (CA). They show that adding synthetic noise to training data improves the performance on test data with natural noise, that is, text with real-world spelling mistakes, while not hurting the performance on clean data. The authors experiment on neural machine translation where the source languages are German, French, Czech, and the target language is English. We hypothesize that adding the right amount of synthetic noise might as well improve the performance on low-resource languages for our set of downstream tasks. For CI, we first build a character vocabulary out of the most commonly used characters in the training set. We do not add noise to one-letter words and do not apply CSW to the first and last characters of the token.

The advantages of character-level synthetic noise are two-fold: First, the output of the augmentation mostly preserves the original syntactic and semantic labels. This is because the resulting tokens are mostly out of vocabulary words that are quite close to the original word—like a spelling mistake. Second, they are trivial to generate, not requiring any external resources like large language models or syntactic annotations. Finally, they are only constrained by the number of characters, which results in the ability of generating huge numbers of augmented sentences—this, can be an advantage for most downstream tasks.

3.2 Token-Level Augmentation

This category includes methods that perform token-level changes such as adding, replacing, or removing certain tokens. While some preserve the syntax or semantics of the original sentence, the majority does not.

Synonym Replacement. As one of the earliest techniques (Kolomiyets, Bethard, and Moens 2011; Zhang, Zhao, and LeCun 2015; Wang and Yang 2015; Wei and Zou 2019), synonym replacement aims to replace words with their synonyms. A lexicon containing synonymity, like WordNet, or a thesaurus is generally required to retrieve the synonyms. Because most languages do not have such a resource, some researchers (Wang and Yang 2015) exploit special pretrained word embeddings and use k-nearest neighbors (by means of cosine similarity) of the queried word as the replacement. As discussed in Section 2, more recent studies (Kobayashi 2018; Wu et al. 2019; Fadaee, Bisazza, and Monz 2017; Anaby-Tavor et al. 2020; Kumar, Choudhary, and Cho 2020) use contextualized language models such as bidirectional LSTMs or BERT (Devlin et al. 2019) to find related words such that the class of the sentence is still preserved. However, these methods require strong pretrained language models that are generally not available for truly low-resource languages. Furthermore, these methods are mostly applied to sentence classification tasks, where the labels are considered coarse-grained compared with our downstream tasks. Considering these, we use a simplified approach similar to Wang and Yang (2015) and query the randomly chosen token on non-contextualized pretrained embeddings. Most languages we experiment on are morphologically productive, having the out-of-vocabulary word problem. To circumvent this issue we use the subword-level fastText embeddings (Grave et al. 2018).

Random Word. Similar to character-level noise, one can inject a higher level of noise by randomly inserting (RI), deleting (RD) or swapping (RS) tokens. The EDA framework by Wei and Zou (2019) shows the efficiency of these techniques—despite their simplicity—on multiple text classification benchmarks. Following Wei and Zou (2019), we experiment with all techniques except RI. Because our downstream tasks require contextual annotation of tokens, we cannot insert a random word without annotation. Similar to character-level methods, random word augmentation techniques are easy to apply and can produce an extensive amount of synthetic sentences as they are only constrained with the number of tokens. One disadvantage is the inability to preserve the syntactic and semantic labels. For instance, deleting a word may yield an ungrammatical sentence without a valid dependency tree. Therefore random word techniques are not eligible for two of our tasks: dependency parsing and semantic role labeling.

3.3 Syntactic Augmentation

This category consists of more sophisticated methods that benefit from syntactic properties to generate new sentences. The main disadvantages of this category are (i) the need for syntactic annotation, and (ii) being more constrained, namely, not being able to generate as many data points as previous categories.

Nonce. This technique, introduced by Gulordava et al. (2018), aims to produce dummy sentences by replacing some of the words in the original sentence, such that the produced sentence is the syntactic equivalent of the original, while not preserving the original semantics. In more detail, randomly chosen content words (i.e., noun, adjective, and verb) are replaced with words that have the same part of speech tag, morphological tags, and dependency label. Given the sentence *"Her sibling bought a cake"*, the following sentences can be generated: *"**My** sibling **saw** a cake"* or *"His **motorbike** bought a **island**"*. As can be seen, the generated sentences mostly do not make any sense, however syntactically equivalent. For this reason, it can only be utilized for syntactic tasks and not for SRL.

Crop. This augmentation algorithm by Şahin and Steedman (2018) morphs the dependency trees to generate new sentences. The main idea is to remove some of the dependency links to create simpler, shorter, but still (mostly) meaningful sentences. In order to do so, Şahin and Steedman (2018) define a list of dependency labels, which is referred to as Label Of Interest (LOI), that attaches subjects and direct and indirect objects to the predicates. Then a simpler sentence is created by retaining only one LOI at a time. Following Şahin and Steedman (2018), we *only* consider the dependency relations that attach subjects and objects to predicates as LOIs, and ignore other adverbial dependency links involving predicates. We keep the augmentation non-recursive, that is, we only reorder the first level of flexible subtrees; hence the flexible chunks inside these subtrees are kept fixed. The idea is demonstrated in Figure 1.

Rotate. This uses a similar idea to *image rotation*, where the sentence is *rotated* around the `root` node of the dependency tree. The method uses the same LOIs as cropping and creates flexible chunks that can be reordered in the sentence. A demonstration of rotation is shown in Figure 1. Both the cropping and rotation operations may cause semantic shifts and ill-formed sentences, mostly depending on the morphological properties of the languages. For instance, languages that use case marking to mark subjects and objects can still generate valid sentence forms via the rotation operation simply because

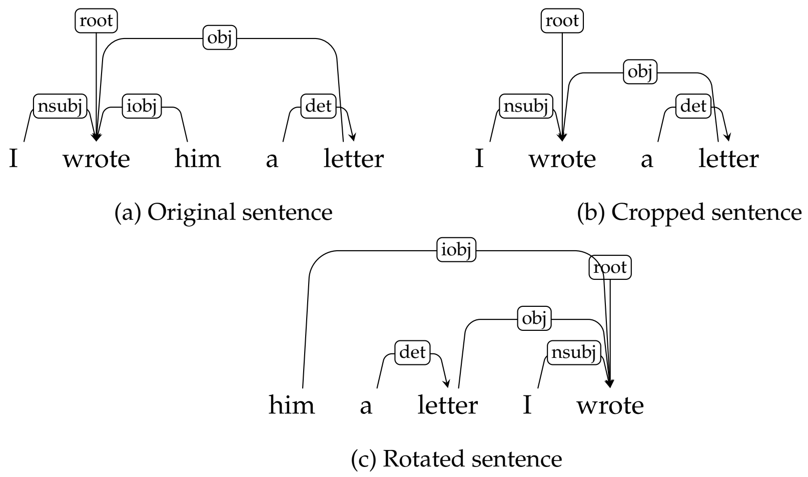

Figure 1
Demonstration of the syntactic augmentation techniques *Crop* and *Rotate*.

word order is more flexible (Futrell, Mahowald, and Gibson 2015). However, most valid sentences may still sound strange to a native speaker, since there is still a *preferred order* and the generated sentence may have an infrequent word order. Furthermore, for languages with a strict word order, like English, rotation may introduce more noise than desired. Finally, the augmented sentences may still be beneficial for learning, since they would provide the model with more examples of variations in semantic argument structure and in the order of phrases.

3.4 Parameters

All of the aforementioned augmentatiom methods are *parametric*. As we show later in Section 5, the choice of parameters may have a substantial impact on the success of the methods. The parameters and their value range are defined as below:

Ratio. This parameter is used to calculate the number of new sentences to be generated from one original sentence. To ensure that more sentences are generated from longer sentences, it is simply calculated as $ratio * |sentence|$; thus, for a sentence of length 10, only 1 extra sentence will be produced with $ratio = 0.1$. It is only used for orthographic methods because syntactic methods are constrained in other ways (e.g., number of tokens that share morphological features and dependency labels). The values we experiment with are as follows: $[0.1, 0.2, 0.3, 0.4]$.

Probability. This determines the probability of a token-level augmentation, namely, the ratio of the augmented tokens to sentence length. For word insertion, deletion, and all character-based methods (CI, CD, CSU, CSW, CA), it can be interpreted as the ratio of tokens that undergo a change to the number of total tokens; whereas for the word swapping operation, it refers to the ratio of swapped token pairs to total number of tokens. Moreover, it determines the number of characters to which the augmentation is applied in cases of character-level augmentation. Similar to ratio, we experiment with the value range: $[0.1, 0.2, 0.3, 0.4]$. For the syntactic methods CROP and ROTATE, probability is associated with the operation dynamics, that is, when the operation probability is set to 0.2, the expected number of additional sentences produced via cropping would be $\#LOI/5$, where $\#LOI$ indirectly refers to the number of valid subtrees for the operation. Because the number of additional sentences are already constrained with $\#LOI$, we also experiment with higher probabilities for *Crop* and *Rotate*. The range is then $[0.1, 0.3, 0.5, 0.7, 1.0]$.

Max Sentences. In cases of considerably long sentences or a substantial number of available candidates, the number of augmented sentences can grow rapidly. This sometimes causes an undesired level of noise in the training data. To avoid this, the number of maximum sentences, namely, sentence threshold parameter, is commonly used. Following Gulordava et al. (2018) and Vania et al. (2019), we experiment with 5 and 10 as the threshold values.

4. Experimental Setup

First we provide detailed information on the downstream tasks and the languages we have experimented on. Next we discuss the data sets that are used in our experiments.

4.1 Tasks

We focus on three sequence tagging tasks that are central to NLP: POS-tagging (POS), dependency parsing (DEP), and semantic role labeling (SRL). POS and DEP require processing at the morphological and syntactic level and are considered crucial steps toward understanding natural language. Furthermore, the performance on POS and DEP tasks are likely to repeat for other tasks that require syntactic and morphological information such as data-to-text generation. On the other hand, SRL is more useful to gain insights on the augmentation performance where preserving the semantics of the original sentence is essential. Therefore it can serve as a proxy for higher-level tasks that necessitate semantic knowledge such as question answering or natural language inference.

The aim of this study is to analyze the performance of various augmentation techniques for low-resource languages in a systematic way, rather than implementing state-of-the art sequence taggers, which would generally require additional resources and engineering effort. Furthermore, this study involves a large number of experiments comparing 11 augmentation techniques across three tasks and several languages in a multi-run setup. For these reasons, we have chosen models that are not heavily engineered and are modular, but also with proven competence on a diverse set of languages. For all tasks, we use neural models that operate on a subword level rather than a word level. This is necessary to tackle the out-of-vocabulary problem, which is inevitable for languages with productive morphology, especially in low-resource scenarios. Furthermore, we do not use any additional features (e.g., POS tags, pretrained word embeddings), external resources, or ensemble of multiple models. Considering these facts, for our POS tagging experiment, we use the subword-level sequence tagging model (Heinzerling and Strube 2019) that is both modular and provides results on par with state-of-the-art on many languages. The model uses an autoregressive architecture (e.g., RNN or bi-LSTM) for random or non-contextual subwords, and uses a fine-tuning paradigm for large pretrained language models. For our dependency parsing experiments, we use the transition-based Uppsala parser v2.3 (de Lhoneux, Stymne, and Nivre 2017; Kiperwasser and Goldberg 2016), which achieved the second best average LAS performance[1] on low-resource languages on the CoNLL 2018 shared task (Zeman et al. 2018). Additionally, we experiment with a biaffine dependency parser built on top of large pretrained contextualized word representations (Glavas and Vulic 2021), since fine-tuning such models on downstream tasks has recently achieved state-of-the-art results on many tasks and languages. Finally, we use a character-level bidirectional LSTM model (Şahin and Steedman 2018) to conduct SRL experiments.[2] More details on the models and their configuration are given in the following subsections.

4.1.1 POS Tagging. The goal is to associate each token with its corresponding lexical class, that is, syntactic label (e.g., *noun, adjective, verb*). Although it is a token-level task, disambiguation using contextual knowledge is mostly necessary as one token may belong to multiple classes (e.g., to fly *[Verb]* or a fly *[Noun]*). For languages with

1 The best performing model (Rosa and Marecek 2018) is a crosslingual model that trains a single model together using multiple treebanks, which was not applicable to our scenario.
2 To the best of our knowledge, there does not exist an open-source dependency-based SRL model built on top of large contextualized language models with proven competence on a diverse set of languages, and implementing such a model is beyond the scope of this article.

rich morphology, it is generally referred to as morphological disambiguation, while the correct morphological analysis—including the POS tag—is chosen among multiple analyses. For analytic languages like English, it is mostly performed as the first step in the traditional NLP pipeline. For this task, we inherit the universal POS tag set that is shared among languages and defined within the scope of the Universal Dependencies project (Zeman, Nivre, and Abrams 2020).

Subword Units. We experiment with three subword units: characters, BPEs, and Word Pieces. Characters and character *n*-grams have been one of the most popular subword units (Ling et al. 2015), because they (i) don't require any preprocessing, (ii) are language-agnostic, and (iii) are computationally cheap due to the small vocabulary size. Byte Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2016b) is a simple segmentation algorithm that learns a subword vocabulary by clustering the frequent character pairs together for a predefined number of times. The algorithm only requires raw text—hence is language-agnostic, and computationally simple. Furthermore, it has been shown to improve the performance of various NLP tasks, especially machine translation. Heinzerling and Strube (2018) trained BPE embeddings using the GloVe (Pennington, Socher, and Manning 2014) word embedding objective and made it available for 275 languages with multiple vocabulary sizes. However, neither randomly initialized character embeddings nor GloVe-BPE embeddings are aware of context. Therefore, we also experiment with contextualized embeddings (i.e., different embeddings for the same subword depending on the context) that operate on Word Pieces. For this study we choose BERT (Devlin et al. 2019), a transformer (Vaswani et al. 2017) based contextualized language model that recently led to state-of-the-art results for many languages and tasks. We use the publicly available multilingual BERT (mBERT)[3] that has been trained on the top 100 languages with the largest Wikipedia using a shared vocabulary across languages. Some of the low-resource languages we use in our experiments are not part of mBERT's languages.

The Model. The overall architecture of the sequence tagging model used in this study (Heinzerling and Strube 2019) is given in Figure 2. Even though the modular architecture enables combining different subwords, we experiment with the units separately for the sake of measuring the individual effect. For the character- and BPE-level model, first each word is split into a sequence of subwords produced by the ρ function. For characters, each subword unit is randomly initialized, whereas for BPE, pretrained embeddings are looked up.

$$\rho(w) = sub_0, sub_1, .., sub_n \tag{1}$$

For character- and BPE-level models, the sequence is encoded via an RNN and a bi-LSTM network, respectively. For the BERT-based model, the encoder is simply the pre-trained transformer that is fine-tuned during training; therefore no additional encoding is performed. For the character-level model, the *final* hidden states are used to represent

3 `https://github.com/google-research/bert/blob/f39e881/multilingual.md`.

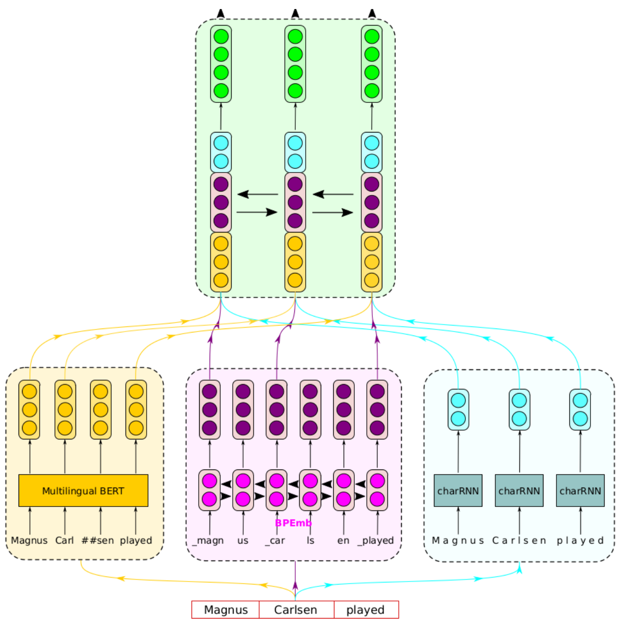

Figure 2
General architecture of the sequence tagger taken from Heinzerling and Strube (2019).

the token, whereas for BPE and mBERT, only the states of the first subword at each token are used.

$$\vec{hs_f}, \vec{hs_b} = \text{Encoder}(\rho(w)) \tag{2}$$

$$\vec{w} = [\vec{hs_f}; \vec{hs_b}] \tag{3}$$

Next, the token embeddings, $\vec{w}$, are passed onto another bi-LSTM layer, where i denotes the index of the word.

$$\vec{h_f}, \vec{h_b} = \text{bi-LSTM}(\vec{w_i}) \tag{4}$$

Then the concatenated hidden states from forward, h_f, and backward directions, h_b, are fed to a classification layer to project the feature space onto the label space. Finally, the probability distribution of each POS tag is calculated via a softmax function. The label with the highest probability is assigned to the input token as follows, where S represents the sentence and $\vec{L}$ denotes the POS tag sequence.

$$p(\vec{L_i}|S) = softmax(W_l \cdot [\vec{h_f}; \vec{h_b}] + \vec{b_l}) \tag{5}$$

BERT is fine-tuned during training, that is, the model's weights are updated by backpropagating through all layers. For each language, we use the best vocabulary size reported in previous work (Heinzerling and Strube 2019). For each subword-level model, we use the parameters from the original work (Heinzerling and Strube 2019).

4.1.2 Dependency Parsing. Dependency parsing aims to provide a viable structural or grammatical analysis of a sentence by finding the links between the tokens. It assumes that the dependent word is linked to its parent, that is, head word, with one of the dependency relations such as *modifier, subject*, or *object*. The resulting grammatical analysis is called a dependency graph, shown in Figure 1. We use the universal dependency label sets defined by the Universal Dependencies project (Zeman, Nivre, and Abrams 2020) and report the Labeled Attachment Score (LAS) as the performance measure. For the experiments, we use two different models: uuparser (Kiperwasser and Goldberg 2016; de Lhoneux, Stymne, and Nivre 2017) and the biaffine parser fine-tuned on large contextualized language models (Glavas and Vulic 2021).

uuparser. The parser is based on the transition-based one of Kiperwasser and Goldberg (2016) that uses bi-LSTMs to create features. Here c refers to character and e refers to embedding. First, a character-based representation is generated via bi-LSTMs. Next, the non-contextual representation for a token is created by concatenating an embedding for the token itself, which we initialize randomly.

$$\vec{e_c} = \text{bi-LSTM}(c_0, c_1, .., c_n) \tag{6}$$

$$\vec{w} = [\vec{e_w}; \vec{e_c}] \tag{7}$$

Then we create a context-aware representation for the token at index i using an additional bi-LSTM layer:

$$\vec{h_i} = \text{bi-LSTM}(\vec{w_i}) \tag{8}$$

We use the arc-hybrid transition-based parser that is later extended with partially dynamic oracle and SWAP transition to be able to create non-projective dependency graphs. A typical transition-based parser consists of so-called configurations and a set of transitions, that is, actions, which can be applied to a configuration to create a new configuration. Parsing starts with a predefined initial configuration. At each iteration, a classifier chooses the best transition given the features extracted from the configuration, and updates the configuration. This step is repeated until a predefined terminal configuration. The arc-hybrid parser used in this work defines configuration as a stack, a buffer, and a set of dependency arcs. Then the feature for the configuration is created by concatenating the representations of a fixed number of tokens calculated via Equation 8 from the top of the stack and the first element of the buffer. Finally, a scoring function that is implemented as a multilayer perceptron assigns scores to transitions and dependency labels, given the extracted feature. The transition and the label with the highest score is then used to create the next configuration. We train separate models for each treebank, using *only* randomly initialized character and word features.

biaffine. The parser consists of an attention layer on top of the outputs from a transformer-based language model, as shown in Figure 3. If a token consists of multiple subword segments, one token representation is created by averaging the transformer outputs for each segment, denoted with $\mathbf{X}$. $\mathbf{X} \in \mathbb{R}^{N \times H}$ is then used as the representation of syntactic dependents, where N and H refer to the number of tokens in the sentence and the transformer hidden state size. To represent the `root` node, the transformer representation for the `[CLS]`: $\mathbf{x}_{CLS}$, that is, the sentence start token, is used. To represent the dependent heads, $\mathbf{X}' = [\mathbf{x}_{CLS}; \mathbf{X}] \in \mathbb{R}^{(N+1) \times H}$ is defined. The arc and relation scores are then calculated as biaffine products as following:

$$\mathbf{Y}_{arc} = \mathbf{X}\mathbf{W}_{arc}\mathbf{X'}^{\top} + \mathbf{B}_{arc}; \mathbf{Y}_{rel} = \mathbf{X}\mathbf{W}_{rel}\mathbf{X'}^{\top} + \mathbf{B}_{rel}$$

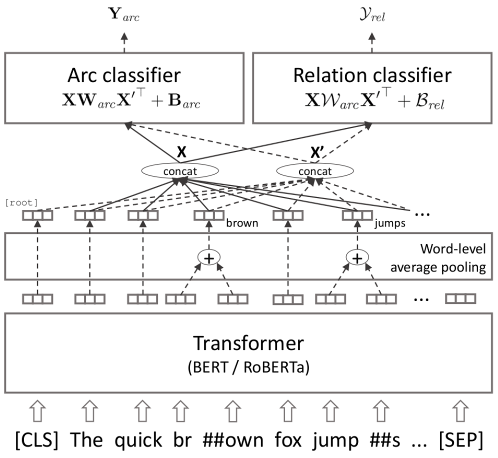
$\mathbf{Y}_{arc}$
$\mathcal{Y}_{rel}$
Arc classifier
$\mathbf{X}\mathbf{W}_{arc}\mathbf{X}'^{\top} + \mathbf{B}_{arc}$
Relation classifier
$\mathbf{X}\mathcal{W}_{arc}\mathbf{X}'^{\top} + \mathcal{B}_{rel}$
X
X'
concat
concat
[root]
brown
jumps
Word-level average pooling
Transformer
(BERT / RoBERTa)
[CLS] The quick br ##own fox jump ##s ... [SEP]

Figure 3
Transformer based architecture of the biaffine parser taken from Glavas and Vulic (2021).

where **W** and **B** refer to the weight and bias parameters for the arc and relation classifiers. The correct dependency head is selected simply as the row corresponding to the maximum score in $\mathbf{Y}_{arc}$. The arc and relation classification losses are defined as cross-entropy losses, respectively, over the sentence tokens and gold arcs.

4.1.3 Semantic Role Labeling. SRL, that is, *shallow semantic parsing*, is defined as analyzing a sentence by means of predicates and the arguments attached to them. A wide range of semantic formalisms and annotation schemes exist; however the main idea is labeling the arguments according to their relation to the predicate.

[I]$_{\text{A0: buyer}}$ [bought]$_{\text{buy.01: purchase}}$ [a new headphone]$_{\text{A1: thing bought}}$ from [Amazon]$_{\text{A2: seller}}$

The example given above shows a labeled sentence with English Proposition Bank (Palmer, Dan, and Paul 2005) semantic roles, where *buy.01* denotes the first sense of the verb "buy", and *A0, A1*, and *A2* are the numbered arguments defined by the predicate's semantic frame. For this study, we perform dependency-based SRL, which means that only the head word of the phrase (e.g., *headphone* instead of *a new headphone*) will be detected as an argument and will be labeled as *A1*. To evaluate SRL results, we used the official CoNLL-09 evaluation script on the official test split. The script calculates the macro-average F1 scores for the semantic roles available in the data.

The Model. Similar to previous models, each token is segmented into subwords. For SRL, we only use characters as the subword unit, because it provides competitive results for many languages (Şahin and Steedman 2018).

$$\rho(w) = sub_0, sub_1, .., sub_n \tag{9}$$

Afterwards, following Ling et al. (2015), a weighted composition of the hidden states, hs_f from forward and hs_b from backward direction, are calculated and used as the token embedding, as given in Equation 11.

$$\vec{hs_f}, \vec{hs_b} = \text{bi-LSTM}(sub_0, sub_1, .., sub_n) \tag{10}$$

$$\vec{w} = W_f \cdot \vec{hs_f} + W_b \cdot \vec{hs_b} + b \tag{11}$$

In order to mark the predicate of interest, we concatenate a predicate flag pf_i to $\vec{w}$ calculated as in Equation 11. It is simply defined as 1 for the predicate of interest and 0 for the rest. Next, $\vec{x_i}$s, are passed onto another bi-LSTM layer, where i denotes the index of the word.

$$\vec{x_i} = [\vec{w}; pf_i] \tag{12}$$

$$\vec{h_f}, \vec{h_b} = \text{bi-LSTM}(\vec{x_i}) \tag{13}$$

The probability distribution of each semantic role label is finally calculated via a softmax function, given the final bidirectional hidden states, h_f and h_b, and the label with the highest probability is assigned to the input token.

$$p(A_i|\vec{S}, P) = softmax(W_l \cdot [\vec{h_f}; \vec{h_b}] + \vec{b_l}) \tag{14}$$

We use the default model parameters reported in Şahin and Steedman (2018).

$$\lambda y\ \lambda x\ add(x, y) \tag{15}$$

$$\lambda y\ \lambda x\ add(x, y)(shoppinglist) \tag{16}$$

$$\lambda x\ add(x, shoppinglist)(ingredient) \tag{17}$$

4.2 Languages

We set the definition of *low-resource language* based on the number of training sentences available in UD v2.6 (Zeman, Nivre, and Abrams 2020). First we calculate the quartiles via ordering the treebanks with respect to their training data size. We then define the languages under the first quartile as low-resource languages, as shown in Figure 4. According to this definition, the list of low-resource languages are as follows: Kazakh, Tamil, Welsh, Wolof, Upper Serbian, Buryat, Swedish sign language, Coptic, Gaelish, Marathi, Telugu, Vietnamese, Kurmanji, Livvi, Belarusian, Maltese, Hungarian, and Afrikaans.

Finally we choose a subset of languages that are from diverse language families: Kazakh (Turkic), Tamil (Dravidian), Buryat (Mongolic), Telugu (Dravidian), Vietnamese (Austro-Asiatic), Kurmanji (Indo-European [IE], Iranian), and Belarusian (IE, Slavic). Although this list of languages can be experimented on for the tasks of POS tagging and dependency parsing, there are no data available for any of these languages for semantic role labeling. Semantically annotated data sets are only available for a handful of languages such as Turkish (Turkic), Finnish (Uralic), Catalan (IE, Romance), Spanish (IE, Romance), and Czech (IE, Slavic). Therefore, we simulate a low-resource environment by sampling for SRL. We provide a summary of languages in Table 3 and discuss the relevant properties of each language family below.

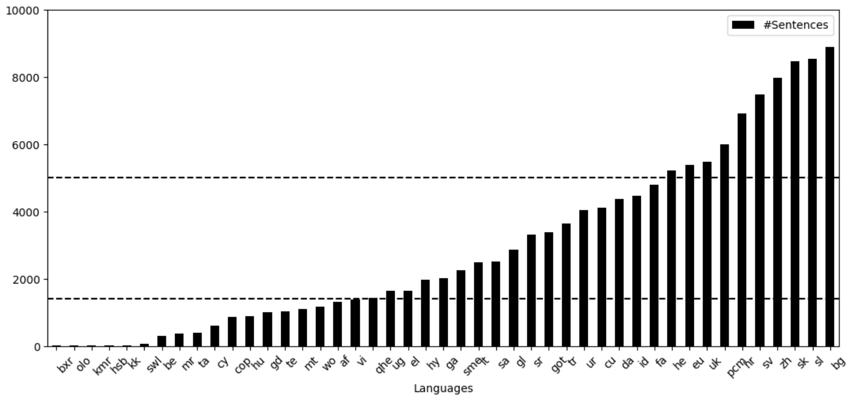

Figure 4
#Training sentences per UD treebank v2.6. Languages are represented with ISO codes. The first line marks the first quartile and second marks the median. Languages with more than 10K training sentences are not shown.

Table 3
The list of languages together with their corresponding language and typological families.

Language	Family	Typology
Belarusian	Indo-European (IE), Slavic	Fusional
Buryat	Mongolic	Agglutinative
Catalan	Indo-European (IE), Romance	Fusional
Czech	Indo-European (IE), Slavic	Fusional
Finnish	Uralic	Agglutinative
Kazakh	Turkic	Agglutinative
Kurmanji	Indo-European (IE), Iranian	Fusional
Spanish	Indo-European (IE), Romance	Fusional
Tamil	Dravidian	Agglutinative
Telugu	Dravidian	Agglutinative
Turkish	Turkic	Agglutinative
Vietnamese	Austro-Asiatic	Analytic

Indo-European (IE). We have five representatives for this language family: Kurmanji, Belarusian, Catalan, Spanish, and Czech. The representative languages are from various branches: Iranian, Slavic, and Romance. From the typological perspective, all languages have fusional characteristics. In other words, morphemes (prefixes, suffixes) are used to convey some linguistic information such as gender; however, one morpheme can be used to mark multiple properties. Slavic languages are known to have around 7 distinct case markers, while others are not as rich. The number and type of case markers available in Slavic helps to relax the word order.

Uralic and Turkic. The languages from both families are known to be agglutinative, meaning that there is one-to-one mapping between morpheme and meaning. All representative languages, namely, Kazakh (Turkic), Turkish (Turkic), Hungarian (Uralic), and Finnish (Uralic), attach morphemes to words extensively. These languages have a high morpheme-to-word ratio, and a comprehensive case marking system.

Table 4
Data set statistics for all tasks, languages, and splits. #sampled: Ranges between 250 and 1,000.
*: No original development set.

		#training	#dev	#test
SRL	Czech	#sampled	5,228	4,213
	Catalan	#sampled	1,724	1,862
	Spanish	#sampled	1,655	1,725
	Turkish	#sampled	844	842
	Finnish	#sampled	716	648
POS & DEP	Vietnamese	1,400	800	800
	Telugu	1,051	131	146
	Tamil	400	80	120
	Belarusian	319	65	253
	Kazakh*	23	8	1,047
	Kurmanji*	15	5	734
	Buryat*	14	5	908

Dravidian. Two languages, namely, Tamil and Telugu, are from the Dravidian language family but in different branches. Similar to Uralic and Turkic, Dravidian languages are also agglutinative, and have extensive grammatical case marking (e.g., Tamil defines eight distinct markers). Unlike other language families, it is not based on the Latin alphabet.

Austro-Asiatic. Vietnamese is the only representative of this family. Unlike the previous languages, Austro-Asiatic languages are analytic. For instance, Vietnamese does not use any morphological marking for tense, gender, number, or case (i.e., it has low morphological complexity).

Mongolic. Being a language from the Mongolic family, Buryat is an agglutinative language with eight grammatical cases. Modern Buryat uses an extended Cyrillic alphabet.

4.3 Data Sets

We use the Universal Dependencies v2.6 treebanks (Zeman, Nivre, and Abrams 2020) for POS and DEP. Some of the languages, such as Kazakh, Kurmanji, and Buryat, do not have any development data. For those languages, we randomly sample 25% of the training data to create a development set. For the SRL task, we use the data sets distributed by Linguistic Data Consortium (LDC) for Catalan (CAT) and Spanish (SPA).[4] In addition, we use dependency-based annotated SRL resources released for Finnish (FIN) (Haverinen et al. 2015) and Turkish (TUR) (Şahin and Adali 2018; Şahin 2016; Sulubacak and Eryiğit 2018; Sulubacak, Eryiğit, and Pamay 2016). All proposition banks are derived from a language-specific dependency treebank, and contain semantic role annotations for verbal predicates. We provide the basic data set statistics for each language in Table 4. Because SRL resources are not available for truly low-resource languages, we sample a small training data set from the original ones, shown with

4 Catalog numbers are as follows: LDC2012T03 and LDC2012T04.

#sampled in Table 4. Each SRL data set uses a language-specific dependency and semantic role annotation scheme. Therefore, we needed to perform language-specific preprocessing for cropping and rotation augmentation techniques given in the Appendix.

5. Experiments and Results

We perform experiments on three downstream tasks, POS, DEP, and SRL, using the models described in Section 4.1. We run each experiment 10 times using the same set of random seeds for the model that operates on the original (unaugmented) and augmented data sets. We compare augmentation techniques to their corresponding unaugmented baselines using a paired t-test and report the p-values. We use dark green for $p \leq 0.05$ and light green for $0.05 < p \leq 0.1$ to highlight the cases where the improvements over the baselines are statistically significant. Similarly we use dark red and light red to denote significantly lower scores. Additionally, we use the sign (**) for $p \leq 0.05$ and (*) for $0.05 < p \leq 0.1$ to highlight the cases where the improvements over the baselines are statistically significant, and similarly the symbols (††) and (†) to denote significantly lower scores. Color-blind friendly results are additionally given in the Appendices.

5.1 POS Tagging

POS tagging is considered one of the most fundamental NLP tasks, such that it has been the initial step in the traditional NLP pipeline for quite a long time. Even in the era of end-to-end models, it has been shown to improve the performance of the downstream tasks of higher complexity when employed as a feature. Taking the importance of POS tagging on higher-level downstream tasks into account, we conduct experiments with different subword units and training regimes (see Section 4.1 for details) to examine (i) whether the behavior of the augmentation methods are model-agnostic, and (ii) whether the improvements are relevant for state-of-the-art models that are fine-tuned on large pretrained multilingual contextualized language models. The results for the character (`char`), BPEs (`BPE`), and multilingual BERT (`mBERT`) are given in Table 5. The languages that lack the corresponding pretrained model (e.g., Kurmanji and `BPE`) are not shown in the table.

char. Except for Telugu, at least one of the techniques has significantly improved over the baselines for all other languages. The token-level methods, *RWD* and *RWS*, have increased the scores of Kazakh and Tamil significantly, while providing a slight increase for Belarusian and Kurmanji. On the other hand, *SR* results in a mixture of increase and decrease in the scores. For instance, Buryat and Kazakh POS taggers benefit significantly from *SR*, while the opposite pattern is observed for Belarusian, Kurmanji, and Telugu. Unlike token-level methods, the character-level ones, *CI*, *CSU*, *CSW*, *CD*, and *CA*, seem to bring more consistent improvements. In particular, Belarusian, Kazakh, and Tamil POS taggers are significantly improved by most of the character-level techniques, whereas the Vietnamese tagger only benefited from *CSU* and *CA* and is slightly hurt by *CD*. In the majority of the cases, syntactic methods either slightly decrease the performance or have not improved significantly. One exception is the *Nonce* technique, which led to a significant improvement for Kazakh, and slight improvements for Buryat and Tamil.

Table 5
Part-of-speech tagging results on original (Org) and augmented data sets, where the results of `char` are given at the top, `BPE` in the middle, and `mBERT` at the bottom.

			Token Level			Character Level					Syntactic		
		Org	**RWD**	**RWS**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**	**Nonce**
`char`	**be**	90.29 ± 1.23	90.41 ± 1.68	91.38 ± 1.36 *	88.57 ± 2.48 †	91.00 ± 0.43 **	91.66 ± 0.39 **	90.24 ± 1.17	91.60 ± 0.79 **	91.50 ± 0.63 **	88.75 ± 2.30 †	90.06 ± 1.12	89.68 ± 1.26
	bxr	39.29 ± 4.16	43.77 ± 6.37	41.48 ± 6.13	46.44 ± 3.16 **	41.36 ± 5.60 *	41.75 ± 5.60	41.15 ± 6.19	41.06 ± 4.83 *	39.99 ± 5.33	43.96 ± 1.53 *	42.37 ± 5.20	42.45 ± 3.57 *
	kk	46.24 ± 3.02	49.65 ± 2.75 **	46.36 ± 3.47	51.25 ± 3.04 **	49.94 ± 2.91 **	50.61 ± 3.15 **	49.51 ± 3.47 **	48.38 ± 4.0 *	51.49 ± 1.22 **	42.57 ± 9.04	46.94 ± 2.84	51.45 ± 0.58 **
	ku	57.04 ± 3.98	57.70 ± 5.20	58.30 ± 3.25 *	56.28 ± 3.71 †	58.90 ± 3.41 *	57.97 ± 3.11 *	56.99 ± 3.20	56.62 ± 3.40	58.60 ± 2.10 *	57.67 ± 3.71 *	54.46 ± 6.77	57.09 ± 2.68
	ta	84.30 ± 1.87	86.96 ± 0.54 *	86.37 ± 0.26 **	85.94 ± 0.35	86.29 ± 0.40 **	86.62 ± 0.14 **	86.37 ± 0.57 **	86.24 ± 0.62 *	86.33 ± 0.77 *	84.63 ± 0.79	84.36 ± 0.57	85.98 ± 0.80 *
	te	90.96 ± 1.38	90.07 ± 1.85	90.51 ± 1.21	90.80 ± 0.36 †	89.90 ± 1.54	90.21 ± 1.23	90.49 ± 1.68	90.62 ± 1.40	90.23 ± 1.73	89.99 ± 0.49 ††	90.01 ± 0.63 †	–
	vi	77.62 ± 0.54	77.45 ± 0.75	77.10 ± 0.90	77.93 ± 0.21	76.97 ± 0.87	77.88 ± 0.24	77.29 ± 0.25	76.00 ± 0.57 †	78.08 ± 0.76	77.17 ± 0.26 †	77.20 ± 0.45 †	–
`BPE`	**be**	88.43 ± 0.59	89.75 ± 0.42 **	89.72 ± 0.38 **	87.61 ± 0.80 †	86.06 ± 1.64 **	88.73 ± 1.68	87.30 ± 1.59 †	88.41 ± 1.47	89.23 ± 1.53 *	87.83 ± 0.74	87.81 ± 1.81	88.40 ± 1.66
	bxr	33.11 ± 0.80	44.21 ± 7.06 **	41.50 ± 7.81 **	46.67 ± 7.55 **	41.82 ± 7.37 **	43.49 ± 7.72 **	39.14 ± 5.37 **	46.06 ± 7.43 **	42.20 ± 6.91 **	28.60 ± 9.77	40.47 ± 7.43 **	44.21 ± 5.51 **
	kk	46.20 ± 0.68	50.36 ± 1.75 **	46.83 ± 0.30 *	50.29 ± 1.66 **	48.64 ± 2.84 *	47.14 ± 0.52 **	48.33 ± 1.84 **	47.28 ± 2.57	47.90 ± 2.40 *	47.02 ± 6.51	49.00 ± 2.16 **	50.75 ± 2.31
	ta	82.76 ± 0.41	81.96 ± 1.00	82.48 ± 0.59	81.51 ± 1.22	82.77 ± 0.31	82.79 ± 0.35	82.22 ± 0.52	82.48 ± 0.49	82.26 ± 0.68	81.69 ± 0.56	80.97 ± 0.64 †	81.97 ± 1.00
	te	88.27 ± 0.60	88.60 ± 0.23 *	88.74 ± 0.63 *	87.71 ± 0.63 ††	88.93 ± 0.26 **	88.21 ± 0.31	87.93 ± 0.67 †	88.27 ± 0.67	88.35 ± 0.34	87.38 ± 0.93	87.68 ± 0.36	–
	vi	77.07 ± 0.46	77.83 ± 0.50 **	77.77 ± 0.33 **	77.83 ± 0.62 **	77.42 ± 0.33	77.39 ± 0.54	77.46 ± 0.56 *	77.53 ± 0.37 *	77.72 ± 0.43 **	75.17 ± 1.08 ††	76.04 ± 0.41 ††	–
`mBERT`	**be**	95.30 ± 0.65	94.50 ± 0.63 †	94.27 ± 0.68 †	94.55 ± 0.92	94.44 ± 1.39	95.70 ± 0.60 **	95.18 ± 0.74	94.76 ± 1.16	95.56 ± 0.34 *	95.64 ± 0.71 *	95.11 ± 0.70	94.48 ± 1.02
	kk	71.34 ± 2.28	70.75 ± 1.89	73.10 ± 1.65 **	71.88 ± 2.16 *	71.65 ± 1.51 *	69.00 ± 4.01	73.11 ± 0.77 **	70.90 ± 1.58	72.57 ± 1.92 *	67.42 ± 2.20	72.00 ± 1.59 *	72.45 ± 1.72 **
	te	89.92 ± 0.48	89.53 ± 0.95	90.12 ± 1.03 *	88.43 ± 0.81 †	89.49 ± 1.17	88.84 ± 1.41	88.59 ± 0.64	89.40 ± 0.41 †	90.15 ± 1.31 *	88.82 ± 1.01 †	89.01 ± 0.48 †	–
	vi	82.41 ± 0.72	81.66 ± 0.97	82.75 ± 0.87	81.04 ± 1.37	82.45 ± 0.89	82.09 ± 1.07	82.48 ± 0.81 *	82.20 ± 0.87	82.57 ± 0.65 *	82.28 ± 0.80	82.33 ± 0.79	–

BPE. In general, BPE scores are slightly lower than char, which can be considered as more room for improvement. Similar to char, we observe improvements over the baselines by at least one technique, with the exception of Tamil. Contrary to char, token-level techniques *RWD* and *RWS* significantly increase the scores for Belarusian, Buryat, Kazakh, Telugu, and Vietnamese. Similar to char, *SR* yields significantly higher scores for Buryat and Kazakh, while significantly reducing the performance for Belarusian and Telugu. Although it brings consistent and significant improvements for Buryat and Kazakh, not many character-level methods lead to higher scores for the other languages. Finally, except for Buryat and Kazakh, the improvements from syntactic techniques are not significant, while the performance drop might be severe in cases like Vietnamese.

mBERT. As expected, the mean baseline scores are the highest for most languages; however, the augmentation methods still provide significant gains in some cases. The most significant improvements are achieved in Belarusian by *CSU* and in Kazakh by *RWS*, *CSW*, and *Nonce*. Unlike char and BPE, we do not observe a distinct increase or decrease pattern in certain groups of techniques, except from *CA*, which achieved significantly higher scores for all languages. However, by manual comparison of the improved/worsened results, the pattern is found to be closer to char than to BPE.

Summary and Discussion. To summarize, we observe significant improvements over the corresponding baselines by a group of augmentation techniques on certain languages for *all* experimented models. The token-level methods provide more significant improvements for BPE models, whereas character-level methods increase the char models' performances the most. Except from *Nonce* and a few exceptional cases, the syntactic-level augmentation techniques either deteriorate the performance or do not lead to significant gains. Even though mBERT baselines are quite strong, it is still possible to achieve significantly better scores—especially for Kazakh, and *CA* leads to consistent improvements across experimented languages. On the other hand, we haven't observed any significant gains by any augmentation for Telugu-char and Tamil-BPE. We also note the following decreasing patterns across models: Belarusian-*SR*, Telugu-*SR*, Telugu-*Crop*, Vietnamese-*Crop*, and Telugu-*Rotate*. We believe the inconsistency of *SR* is due to: (i) pretrained embeddings do not guarantee a syntactic equivalent of a token, and (ii) quality of embeddings also suffer from low-resources, namely, small Wikipedia.

Furthermore, the linguistically motivated syntactic techniques such as *Crop* and *Rotate* are found to be less effective than the straightforward techniques that rely on introducing noise to the system, namely, *RWD*, *RWS*, *CI*, *CSU*, *CSW*, *CD*, and *CA*. One important difference between these two categories is the *amount of augmented data* that can be generated via both techniques. This amount is limited to linguistic factors for syntactic techniques, such as the number of subtrees or the number of tokens that share the same morphological features and dependency labels. On the other hand, the number of sentences with noise addition is only constrained by the parameters. Hence, substantially larger amounts of augmented data can be generated via simple techniques than the more sophisticated ones.

Additionally, we believe that this additional number of noisy sentences provides informative signals to the POS tagging network. As shown previously (Tenney, Das, and Pavlick 2019), low-level features like POS tags are mostly encoded at the initial layers of the network. As layers are added to the network, more sophisticated features that require understanding of the interaction between tokens are more likely to be captured.

This is connected to the nature of the unsophisticated augmentation techniques that treat tokens as isolated items, rather than considering the relation among other tokens like the syntactic methods. As a result, easier techniques yield stronger associations at the initial layers, while more sophisticated techniques are more likely to strengthen the intermediate layers. In addition to easier techniques, the syntactic method *Nonce* also advanced the scores of most POS taggers. Although it is considered one of the sophisticated methods, it targets tokens instead of modifying the sentence structure.

5.2 Dependency Parsing

We use the transition-based parser `uuparser` and the `biaffine` parser based on mBERT from Rust et al. (2021) to conduct the dependency parsing experiments. The mean and the standard deviation for each language-data set pair are given in Table 6.

uuparser. Compared with POS tagging, the relative improvements over the baseline are substantially higher. However, unlike POS tagging there is no linear relation between low baseline scores and larger improvements. For instance, the relative improvement for Kazakh is only 3%, despite having the second lowest baseline. This suggests that more factors such as data set statistics and linguistic properties come into play for dependency parsing. Except for Vietnamese, we observe significant gains for all languages by at least one of the augmentation methods—sometimes by *any* technique as in Kurmanji. Similar to POS tagging, *SR* provides a mixture of results, significantly reducing the scores for Belarus and Vietnamese, while improving the Kurmanji and Buryat parsers. Character-level augmentation techniques mostly improve the performance of Belarusian, Buryat, Kazakh, Kurmanji, and Telugu parsers significantly. Unlike POS tagging, the improvements from syntactic methods are more emphasized for most languages with the exception of Vietnamese, Belarusian, and Buryat (only in Nonce augmentation setting).

biaffine. As discussed earlier, mBERT is trained on the top 100 languages with the largest Wikipedia; however, training is performed on a *shared* vocabulary. Hence even if a language is not seen during training, the shared vocabulary might enable *zero-shot* learning—especially if the model is trained with related languages. In Table 6,

Table 6
Dependency parsing LAS scores on original (Org) and augmented data sets, where the results of `uuparser` are given at the top, and `biaffine` parser at the bottom. *language*ˆ denotes that *language* is part of the multilingual BERT.

			Token Level		Character Level				Syntactic		
		Org	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**	**Nonce**
`uuparser`	**be**	52.85 ± 0.70	50.10 ± 0.67 ††	54.05 ± 0.40 **	53.39 ± 0.51	55.37 ± 1.48 *	54.89 ± 0.87 *	53.57 ± 0.97 *	53.01 ± 1.01	53.03 ± 0.31	54.27 ± 0.68
	bxr	11.73 ± 1.65	12.70 ± 1.84 *	12.39 ± 1.31	13.04 ± 1.20 **	13.83 ± 0.46 **	13.44 ± 1.06 **	14.06 ± 1.14 **	11.99 ± 1.52 **	11.43 ± 1.39	10.70 ± 0.77 ††
	kk	23.82 ± 0.92	23.82 ± 0.98	24.36 ± 0.59	24.80 ± 0.75 **	24.87 ± 0.77 *	24.79 ± 0.86 *	24.62 ± 0.83 *	24.09 ± 0.79	23.76 ± 0.32	24.78 ± 1.07 *
	ku	21.09 ± 0.86	23.22 ± 0.60 **	23.70 ± 0.92 **	24.08 ± 0.74 **	23.48 ± 0.87 *	24.32 ± 0.55 **	24.39 ± 0.12 **	22.74 ± 0.35 **	24.50 ± 0.44 **	23.98 ± 0.31 **
	ta	55.52 ± 0.39	54.27 ± 1.99	55.39 ± 0.42	55.15 ± 0.71	54.89 ± 0.90	56.11 ± 0.77	56.52 ± 1.31	54.62 ± 0.65 *	55.16 ± 0.18 *	57.50 ± 0.43 **
	te	77.79 ± 0.85	76.94 ± 0.94	77.65 ± 0.94	78.83 ± 0.73 *	77.27 ± 1.12 *	78.11 ± 0.80 **	78.87 ± 0.51	78.01 ± 0.86 *	78.27 ± 1.05 *	–
	vi	55.01 ± 0.15	48.23 ± 0.27 ††	53.80 ± 0.32	52.26 ± 0.47 †	52.19 ± 0.36 ††	52.92 ± 0.23 ††	53.15 ± 0.42 †	55.33 ± 0.40	54.87 ± 0.26	–
`biaffine`	**be**ˆ	78.17 ± 0.36	79.66 ± 0.28	80.47 ± 0.11 **	80.04 ± 0.80 **	78.79 ± 0.64	80.58 ± 0.49 **	80.45 ± 0.36 **	79.84 ± 0.30 **	79.78 ± 0.35 **	79.33 ± 0.27 **
	bxr	17.71 ± 0.52	18.73 ± 0.13 **	17.91 ± 0.51	17.65 ± 0.44	17.73 ± 0.17	17.72 ± 0.87	17.81 ± 0.50 *	16.74 ± 0.88 †	17.59 ± 0.36	17.12 ± 0.82
	kkˆ	34.23 ± 0.32	35.49 ± 0.75 **	35.61 ± 0.52 **	36.69 ± 0.76 **	36.30 ±0.59 **	35.42 ± 0.49 **	35.76 ± 0.42 **	34.61 ± 0.75 *	34.18 ± 0.42	38.43 ± 0.17 **
	ku	20.48 ± 0.35	20.64 ± 0.24	22.16 ± 0.47 **	20.49 ± 0.33	22.56 ± 0.68 **	22.71 ± 0.30 **	23.13 ± 0.26 **	20.53 ± 0.41	19.39 ± 0.50 ††	22.82 ± 0.24 **
	taˆ	70.01 ± 0.62	70.99 ± 0.21 **	70.27 ± 0.10	71.21 ± 1.08 *	70.54 ± 0.49	71.13 ± 0.67 **	71.47 ± 0.53 **	70.89 ± 0.30 **	71.11 ± 1.01 *	71.24 ± 0.85 **
	teˆ	84.60 ± 0.79	84.60 ± 0.80	84.28 ± 0.97 †	84.88 ± 0.54 **	85.02 ± 0.32 **	84.05 ± 0.78 ††	84.74 ± 0.81	85.30 ± 0.57 **	84.19 ± 0.59 ††	–
	viˆ	66.25 ± 0.84	62.40 ± 0.76 ††	63.49 ± 0.80 ††	63.87 ± 0.55 ††	64.22 ± 2.07 ††	64.54 ± 2.05 ††	64.09 ± 1.25 ††	66.11 ± 0.85	65.94 ± 0.91	–

the languages without an * sign are not included in mBERT training; therefore the scores are the results of zero-shot learning. Compared with `uuparser`, all baselines (Org) are substantially higher as expected—except for Kurmanji. Despite such high scores, we observe a considerable amount of statistically significant improvements over the baseline with some exceptions. As with the `uuparser`, Vietnamese dependency parsing performance is significantly reduced by augmentation, suggesting that augmentation methods introduce too much noise for the Vietnamese dependency parser. *SR* improves the scores more consistently for Belarus, Kazakh, and Tamil, while not being able to bring significant gains for the other languages. Character noise injection techniques—especially *CD* and *CA*—boost the performance of most parsers, again with the exception of Vietnamese and Telugu. Similar to `uuparser`, syntactic techniques result in higher scores compared to POS tagging. We observe significant gains from *Nonce*, while *Crop* also improves substantially in most cases—Buryat and Vietnamese are exceptions. Unlike *Nonce* and *Crop*, *Rotate* gives mixed results.

Summary and Discussion. The similarities among different parsers and POS taggers are numerous, such as (i) the small number of augmentation techniques that were able to improve scores for Buryat; (ii) the high performance of character noise methods for most languages; (iii) the generally confusing scores produced by the *SR* and *rotation*, and (iv) the mostly unimproved or worse results for Vietnamese and Telugu. Considering the synthetic nature of most languages, where characters (e.g., case markers) provide valuable information on the relationship between two tokens, high performance of character-level noise is rather expected. In other words, varying character sequences may help the network to strengthen the dependency relations. Unlike POS tagging, we also observe a performance boost from the syntactic augmenters *Crop* and *Nonce*, sometimes leading to the best results, for example, for Vietnamese and Kurmanji. This suggests that introducing more syntactic variation/noise, even in smaller amounts than character-level noise, helps in certain cases. Nevertheless, the performance of both techniques is comparable, and it is not possible to single out one technique that is guaranteed to improve the results. Additionally, we observe that not *all* character-level methods increase the scores and there is no clear pattern to which character noise improves which language. Our results also suggest that a higher number of augmentation techniques are able to improve significantly over competitive baseline scores provided by `biaffine` compared to the `mBERT`-based POS tagger. One reason may be that the `mBERT` model already contains a substantial amount of low-level syntactic knowledge and augmentation techniques only add noise to the fine-tuning process.

5.3 Semantic Role Labeling

As discussed in Section 4.1, we *simulate* a low-resource scenario by sampling 250, 500, and 1,000 training sentences from the original training sets. We perform a grid search on augmentation parameters for the #sampled = 250 setting, and choose the parameters with the best average performance for other data set settings. The results are given in Table 7.

As expected, the relative improvement over the baselines decreases, as the number of samples increases. For some languages like Finnish, the drop is dramatic, while for the majority, the decrease is exponential. The only exception is Czech. The reason is the fine-grained, language-specific semantic annotation scheme that requires larger data sets. Another noticeable pattern is the decreasing number of augmentation methods that improve the F1 scores with the increasing sample size. For instance, while six of the

Table 7
Semantic role labeling results on original (Org) and augmented data sets.

					Catalan				
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	31.34 ± 0.99	37.29 ± 1.26 **	38.50 ± 1.37 **	37.74 ± 1.64 **	39.79 ± 0.25 **	37.45 ± 1.58 **	38.28 ± 1.48 **	26.59 ± 0.52 ††	25.96 ± 0.62 ††
500	44.53 ± 1.39	44.12 ± 1.78	49.75 ± 0.10 **	47.77 ± 0.65 *	46.81 ± 1.59 *	47.47 ± 0.62 *	49.46 ± 0.19 **	44.30 ± 1.16 †	44.72 ± 0.36
1,000	53.06 ± 1.33	53.49 ± 0.64	53.80 ± 0.67	54.83 ± 0.81	56.37 ± 0.29 *	55.50 ± 0.17 *	53.97 ± 0.96	53.57 ± 0.07	52.31 ± 0.96
					Turkish				
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	31.28 ± 0.12	31.78 ± 1.20	31.62 ± 1.53	30.67 ± 2.19	30.14 ± 2.46	32.52 ± 0.89 *	32.02 ± 1.53	32.42 ± 0.64 *	30.37 ± 1.75
500	35.75 ± 1.38	35.95 ± 0.65	36.35 ± 0.68	38.27 ± 0.88 *	37.30 ± 1.05	34.80 ± 1.96	36.23 ± 1.37	37.40 ± 1.07 *	34.58 ± 1.89
1,000	44.89 ± 0.80	42.41 ± 1.20	43.36 ± 0.67	41.78 ± 1.39	42.28 ± 1.31	41.42 ± 1.38	29.14 ± 0.47 ††	44.83 ± 1.07	42.71 ± 0.76
					Spanish				
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	31.23 ± 1.30	34.65 ± 1.02 *	36.31 ± 2.28	37.91 ± 1.21 **	36.80 ± 1.90 *	36.76 ± 2.11 *	37.34 ± 1.14 **	26.04 ± 1.34 ††	26.95 ± 2.22 ††
500	44.16 ± 0.68	43.89 ± 0.65	44.80 ± 0.81	44.82 ± 1.10	43.75 ± 1.36	44.60 ± 0.72 *	45.03 ± 0.91 *	42.35 ± 0.42 ††	41.13 ± 0.38 ††
1,000	50.98 ± 1.30	50.80 ± 1.21	53.03 ± 0.76 **	52.53 ± 0.60 **	52.10 ± 1.00 *	52.76 ± 0.57 **	54.12 ± 0.12 **	51.32 ± 1.09	52.40 ± 0.34 *
					Czech				
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	35.63 ± 1.08	33.48 ± 1.58	37.17 ± 0.13 *	37.56 ± 0.02 *	31.56 ± 1.00 †	35.70 ± 0.80	36.14 ± 1.78	34.84 ± 1.55 **	30.78 ± 0.38 ††
500	44.96 ± 0.51	42.04 ± 1.08 †	46.44 ± 1.20 *	46.13 ± 1.97	46.24 ± 1.60	47.92 ± 0.59 **	47.42 ± 1.07 *	46.38 ± 0.12 *	44.83 ± 1.47
1,000	50.29 ± 0.09	48.15 ± 0.31 †	48.66 ± 0.49 †	52.63 ± 0.97 *	51.10 ± 1.23	47.85 ± 1.19 †	51.97 ± 0.54 *	49.33 ± 1.36	48.45 ± 1.34
					Finnish				
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	18.80 ± 0.89	18.60 ± 2.34	17.18 ± 1.56	19.68 ± 1.16	25.71 ± 0.80 **	28.87 ± 1.55 **	27.96 ± 0.44 **	30.83 ± 0.17 **	20.07 ± 2.42
500	37.23 ± 0.35	34.10 ± 1.61	35.59 ± 0.87 †	36.88 ± 0.29 †	38.98 ± 0.03 *	35.29 ± 0.99 †	35.32 ± 1.63	37.58 ± 0.93 *	35.16 ± 1.08
1,000	41.64 ± 0.98	40.15 ± 0.95 †	41.27 ± 0.84	41.41 ± 1.08	39.80 ± 0.81	40.57 ± 1.17	41.99 ± 0.04	41.35 ± 1.37	39.08 ± 0.34 †

methods increase the SRL performance in the #sample = 250 setting for Catalan, only two of them provides improvement for the #sample = 1,000 setting—which are also less significant.

We see one distinctive pattern for the languages Turkish, Czech, and Finnish. Unlike Spanish and Catalan, the syntactic operation *crop* improves the performances for almost all settings. We believe this is due to the rich case marking systems of these languages that enable generating almost natural sentences from subtrees. Furthermore, we observe that the *rotation* operation introduces a high amount of noise that cannot be regularized easily with the models. One reason for more consistent improvements with the cropping operation is related the semantic role statistics. Most treebanks in this study are dominated by the core arguments, namely, Arg0, Arg1. These core arguments are usually observed as subjects or objects. In addition, many predicates are encountered with missing arguments, that is, it is more likely to see a predicate with only one of the arguments than containing all. We believe cropping introduces more variation in terms of core arguments compared to rotation, which provides more signals for the cases such as missing arguments. For Spanish and Catalan, the gains are almost always provided by character-level noise. On the contrary, both languages never benefit from the syntactic operations, as expected.

5.4 Summary of the Findings

We summarize the key findings from experiments from different perspectives: languages, downstream tasks, augmentation techniques, and models.

5.4.1 Languages

- Most languages see significant improvements in at least one augmentation configuration independent of tasks, models, and the data set sizes.
- Vietnamese has mostly witnessed drops in scores, which were especially highlighted in dependency parsing. This suggests that the augmentation methods may be less effective for analytic languages.

- We have not observed any significant difference between fusional and agglutinative languages by means of POS and DEP scores, although the differences between syntactic and non-syntactic augmentation techniques were pronounced for languages with different morphological properties.
- The suitability of augmentation techniques have been found to be dependent on the language and subword unit pair. For instance, Tamil POS tagging with `BPE` baseline could not be improved, where we have observed significant improvements over the Tamil POS tagging `char` baseline.
- We have detected inconsistent results for the Telugu language in the majority of cases. Furthermore, we have not seen many configurations that led to substantial improvements. Telugu had the second largest number of significant declines in performance after Vietnamese. We believe this may be due to Telugu having one of the largest treebanks, and augmentation techniques adding unmeaningful noise to the already strong baseline.

5.4.2 Tasks

- We have found many similarities between the results for POS and DEP. The most important ones are: (i) character-level augmentations providing significant improvements in most cases and (ii) inconsistent results from *SR* and *rotation* methods.
- The task that has been improved the most (i.e., statistically significant improvements with a large gap over the baseline) was DEP, followed by POS and SRL. In other words, the experimented augmentation benefited the task with the *intermediate* complexity the most.
- Strong baseline scores provided by exploiting large pretrained contextualized embeddings were more likely to be further improved for DEP (e.g., `biaffine`) than POS (e.g., `mBERT`).

5.4.3 Augmentation Techniques

- The most consistent augmenters across tasks, models, and languages were found to be the character-level ones.
- A satisfactory choice of augmentation techniques depends heavily on the input unit. For instance, token-level augmentation provides significant improvements for `BPE`, while character-level augmentation gives higher scores for `char` and WordPiece of `mBERT`.
- The performance of *SR* has been detected as irregular. The reason may be that it relies on external pretrained embeddings that may be of lower-quality for some of the languages.
- Even if we have not found one single winner across character-level augmentation methods, the mixed character noise, namely, *CA*, has improved the POS, DEP, and SRL tasks more consistently.

- Among the syntactic augmenters, *crop* and *Nonce* have been found to be more reliable compared with *rotate*—with some exceptions like the case for `BPE`.

5.4.4 Models

- We observed almost a regular improvement pattern among different models for DEP, such as a significant drop in scores for Vietnamese. Even though there were some similarities among models for POS, such as syntactic augmenters achieving the lowest scores, a regular pattern was not visible. The reason might be the difference among the subword units in POS.[5]
- Strong baseline scores provided by exploiting large pretrained contextualized embeddings were more likely to be further improved for DEP (e.g., `biaffine`) than POS (e.g., `mBERT`). This may be due to `mBERT` already containing a substantial amount of low-level syntactic knowledge (e.g., POS), hence augmentation techniques only add noise to the fine-tuning process.

6. Analysis

In this section, we first discuss the parameter choice for augmentation techniques and conduct a case study on POS tagging to analyze their sensitivity to parameter choice and their performance range. Next, we study the performance of individual augmentation techniques on frequent and infrequent tokens and lexical types to reveal whether the performance improvement also corresponds to better generalization in out-of-domain settings.

6.1 Augmentation Parameters

Augmentation methods are parameterized and their performance may be sensitive to changes in parameter values. However in a real-world low-case scenario (mostly) without any development set, parameter tuning may not always be possible. Therefore we create augmented data sets using all combinations of augmentation parameters described in Section 3 (e.g., $4 \times 4 \times 2 = 32$ augmented data sets with RWD method for each treebank). To analyze the behavior of each augmentation technique, we perform a grid search on the parameters and draw the box plots for each augmentation method on Belarusian, Tamil, Vietnamese, Buryat, Kazakh, and Kurmanji POS tagging shown in Figure 5. The box plots ensure that the augmentation techniques can be compared with respect to their performance range and their sensitivity to parameters rather than their best performance.

Belarusian. The median lines of RWD, RWS, CI, CSU, CA, and the Nonce lie outside of the Org (baseline) box. This suggests that these techniques are statistically likely to perform higher than the baseline. Out of these methods, Nonce and RWD performance are found to be less dispersed than the others, that is, are more prone to parameter changes.

5 All POS models use *distinct* input types: character, BPE, and WordPiece; while `uuparser` (using a combination of char and word) and `biaffine` (WordPiece) parsers are likely to share more vocabulary.

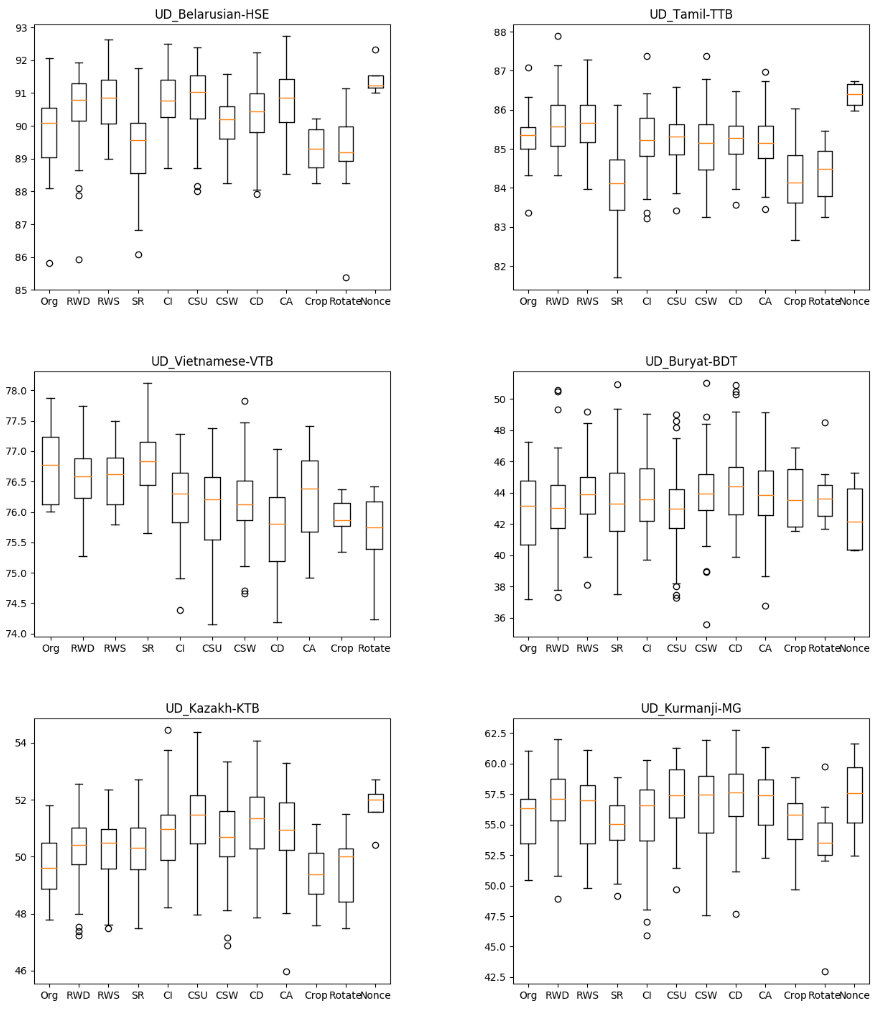

Figure 5
Box plots for augmentation model performances on POS Tagging. Org box refers to the baseline on the original data set, while other techniques are defined in Section 3.

The min-max range of Nonce is quite smaller than others, signaling the reliability of the method. There are only a few outliers to most techniques, suggesting that the results mostly follow a distribution.

Tamil. For Tamil, only RWS and Nonce median lines lie above the baseline box. Similar to Belarusian, the box length and the min-max range of Nonce is smaller than RWS, hinting at the reliability of the model. Unlike Belarusian, SR, crop, and rotate lie under the box, meaning that they are likely to yield worse results than the baseline. Similarly, the number of outliers are limited and the maximum improvement over the best baseline model is around 0.7%.

Vietnamese, Hungarian. For Vietnamese, there are no methods that are significantly better than the baseline, but only worse: CD, crop, and rotate. Although the maximum value of SR surpasses the best baseline model, the box plots reveal that this is statistically unlikely. A similar pattern is observed for Hungarian, despite being from a different language family.

Buryat, Telugu. While the training data set of Buryat is the smallest of all, even modest changes in parameters may lead to outliers. Interestingly, we found none of the techniques to be significantly better or worse than the baseline according to the median lines. However, the outliers provide a performance boost around 8% over the best baseline. Even the plot is not shown for convenience; we noticed that all augmentation techniques, except for SR, provide neither a significant drop nor an increase in the scores for Telugu, similar to Buryat.

Kazakh, Kurmanji. For both languages, the median lines above the baseline box are the character-level methods and the syntactic method Nonce. In Kazakh, similar to Belarusian and Tamil, Nonce has the smallest box and the min-max range. For Kurmanji, however, CA has the smallest box and min-max range instead of Nonce. Kazakh and Kurmanji, as the languages with the lowest resources, benefit significantly more from character-level augmentation compared with other languages. This may be due to a higher ratio of out-of-vocabulary words for these treebanks. In other words, association of a character set to POS labels is more important than association of tokens with POS labels. Apparently, character-level noise helps to strengthen such links.

6.2 Frequency Analysis

Besides improving the downstream task scores, one of the important goals of an augmentation method is to increase the generalization capability of the model. In order to evaluate the extent of this skill, we measure the individual performances on frequent and infrequent tokens along with word types. We perform a case study on dependency parsing because it stands between POS and SRL by means of complexity. The results are given in Figure 6.

Frequent vs. Infrequent Tokens. First, we define the token as *frequent* if it is among the top 10% in the token frequency list extracted from the combination of training and

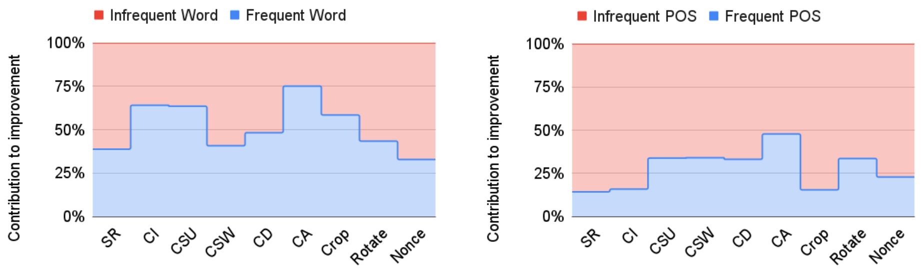

Figure 6
Individual contributions of correct labeling of frequent and infrequent tokens and POS tags to the overall parser performance. Shown separately for each augmentation technique.

development set. Next, we create a vocabulary of frequent tokens and then measure the LAS scores individually for frequent and infrequent tokens for each language and augmentation technique. Finally, we average the scores over all languages to ease the presentation of the results. The results show that only *SR*, *CSW*, *Rotate*, and *Nonce* improve the scores of infrequent tokens than frequent ones. *SR* and *Nonce* are likely to replace frequent tokens with infrequent ones, since their objective is to replace words with another. However, *CSW* and *Rotate* are not designed to replace tokens. We believe character switching sometimes coincidentally resulted in rare tokens, and the *Rotate* operation has randomly chosen the subtrees with rare tokens to augment. Interestingly, there is no one-to-one correspondence between the techniques that improved the dependency scores the most and the techniques that improved labeling of infrequent tokens the most. This may be due to building the vocabulary over tiny training sets and identifying the frequent/rare tokens accordingly. Therefore we analyze a more general property: POS tags.

Frequent vs. Infrequent POS Tags. We perform a similar analysis for the token class—using the gold POS tags as the class. Because the number of unique POS tags is much lower than unique tokens, we identify the top 50% of the POS tags as frequent. We use the same calculation technique as above. The results show that all techniques improve the performance on rare token classes more than the frequent ones; however, there still does not exist a direct correlation between the best performing technique and the improvement over infrequent POS tags. This suggests that the improvement cannot simply be explained by frequency analysis, that is, the method that focuses on improving rare tokens or token classes is not guaranteed to improve the overall score. This is because the parser's performance is a more complicated multivariate variable that relies on many other factors apparent in data set statistics.

7. Conclusion

Neural models have emerged as the standard model for a wide range of NLP tasks, including part-of-speech tagging, dependency parsing, and semantic role labeling—the task of assigning semantic role labels to predicate-argument pairs. These models were shown to provide state-of-the-art scores in the presence of large training data sets, although they still fall behind traditional statistical techniques in genuinely low-resource scenarios. One method that is commonly used to overcome the low data set size problem is enhancing the original data set synthetically, which is referred to as data augmentation. Recently, a variety of text augmentation techniques have been proposed such as replacing words with their synonyms or related words, injecting spelling errors, generating grammatically correct but meaningless sentences, simplifying sentences, and many more.

Despite the richness of augmentation literature, previous work mostly explores text classification and machine translation tasks on high-resource languages, and reports single scores. In this study, on the contrary, we provide a more detailed performance analysis of the existing augmentation techniques on a diverse set of languages and traditional sequence tagging tasks that are more sensitive to noise. First, we compile a rich set of augmentation methods of different categories that can be applied to our tasks and languages, namely, character level, token level, and syntactic level. Then we systematically compare them on the following downstream tasks: part-of-speech tagging, dependency parsing, and semantic role labeling. We conduct experiments on truly low-resource languages (when possible) such as Buryat, Kazakh, and Kurmanji;

and simulate a low-resource setting on a diverse set of languages such as Catalan, Turkish, Finnish, and Czech when data are not available.

We find that the easy-to-apply techniques such as injecting character-level noise or generating grammatical, meaningless sentences provide gains across languages more consistently for the majority of cases. Compared with POS tagging, we have observed more significant improvements over the baseline for dependency parsing. Although the augmentation patterns for dependency parsing are found similar to POS tagging, a few differences, such as larger gains from syntactic methods, are noted. Again, the largest improvements in dependency parsers are mostly obtained by injecting character-level noises. For SRL, we show that the improvement from augmentation decreases with more training samples, along with the number of augmentation techniques that increases the scores. We observe that languages with fusional morphology almost always benefit from character-level noise the most; but always suffer from the syntactic operations *crop* and *rotate*. On the contrary, agglutinative languages such as Turkish and Finnish benefit from cropping, while character-level augmentation may provide inconsistent improvements. We show that augmentation techniques can provide consistent and statistically significant improvements for all languages except Vietnamese and Telugu. We find that the improvements do not solely depend on the task architecture, that is, augmentation methods can further improve on the strong `biaffine` parser as well as the weaker `uuparser`.

Acknowledgments

We first thank the anonymous reviewers who helped us improve the article. We would like to thank Clara Vania, Benjamin Heinzerling, Jonas Pfeiffer, and Phillip Rust for the valuable discussions and providing early access to their implementations. We finally thank Celal Şahin, Necla İşgüder, and Osman İşgüder for their invaluable support during the writing of this article.

References

Anaby-Tavor, Ateret, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? Deep learning to the rescue! In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pages 7383–7390. `https://doi.org/10.1609/aaai.v34i05.6233`

Andreas, Jacob. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566. `https://doi.org/10.18653/v1/2020.acl-main.676`

Belinkov, Yonatan and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, pages 1–13.

Chen, Hannah, Yangfeng Ji, and David Evans. 2020. Finding friends and flipping frenemies: Automatic paraphrase data set augmentation using graph theory. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4741–4751. `https://doi.org/10.18653/v1/2020.findings-emnlp.426`

Chen, Jiaao, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021. An empirical survey of data augmentation for limited data learning in NLP. *CoRR*, abs/2106.07499:1–19.

Chen, Jiaao, Zhenghui Wang, Ran Tian, Zichao Yang, and Diyi Yang. 2020a. Local additivity based data augmentation for semi-supervised NER. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 1241–1251. `https://doi.org/10.18653/v1/2020.emnlp-main.95`

Chen, Jiaao, Zichao Yang, and Diyi Yang. 2020b. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157. `https://doi.org/10.18653/v1/2020.acl-main.194` de Lhoneux, Miryam, Sara Stymne, and Joakim Nivre. 2017. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies, IWPT 2017*, pages 99–104.

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

Ding, Bosheng, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057. `https://doi.org/10.18653/v1/2020.emnlp-main.488`

Fadaee, Marzieh, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 2: Short Papers*, pages 567–573. `https://doi.org/10.18653/v1/P17-2090`

Feng, Steven Y., Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. GenAug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42. `https://doi.org/10.18653/v1/2020.deelio-1.4`

Feng, Steven Y., Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard H. Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, pages 968–988. `https://doi.org/10.18653/v1/2021.findings-acl.84`

Feng, Steven Y., Aaron W. Li, and Jesse Hoey. 2019. Keep calm and switch on! Preserving sentiment and fluency in semantic text exchange. 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 2701–2711.

Futrell, Richard, Kyle Mahowald, and Edward Gibson. 2015. Quantifying word order freedom in dependency corpora. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 91–100.

Gao, Fei, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544. `https://doi.org/10.18653/v1/P19-1555`

Glavas, Goran and Ivan Vulic. 2021. Is supervised syntactic parsing beneficial for language understanding tasks? An empirical investigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021*, pages 3090–3104. `https://doi.org/10.18653/v1/2021.eacl-main.270`

Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, pages 3483–3487.

Grundkiewicz, Roman, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263. `https://doi.org/10.18653/v1/W19-4427`

Gulordava, Kristina, 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, NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 1195–1205. `https://doi.org/10.18653/v1/N18-1108`

Guo, Demi, Yoon Kim, and Alexander M. Rush. 2020. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 5547–5552. `https://doi.org/10.18653/v1/2020.emnlp-main.447`

Guo, Hongyu. 2020. Nonlinear mixup: Out-of-manifold data augmentation for text classification. In *The Thirty-Fourth*

AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, pages 4044–4051. https://doi.org/10.1609/aaai.v34i04.5822

Guo, Hongyu, Yongyi Mao, and Richong Zhang. 2019a. Augmenting data with mixup for sentence classification: An empirical study. *CoRR.*, abs/1905.08941:1–7.

Guo, Hongyu, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 3714–3722. https://doi.org/10.1609/aaai.v33i01.33013714

Han, Wenjuan, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020. Adversarial attack and defense of structured prediction models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 2327–2338. https://doi.org/10.18653/v1/2020.emnlp-main.182

Haverinen, Katri, Jenna Kanerva, Samuel Kohonen, Anna Missila, Stina Ojala, Timo Viljanen, Veronika Laippala, and Filip Ginter. 2015. The Finnish Proposition Bank. *Language Resources and Evaluation*, 49(4):907–926. https://doi.org/10.1007/s10579-015-9310-y

Hedderich, Michael A., Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, pages 2545–2568. https://doi.org/10.18653/v1/2021.naacl-main.201

Heinzerling, Benjamin and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018*, pages 2989–2993.

Heinzerling, Benjamin and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, pages 273–291.

Jindal, Amit, Arijit Ghosh Chowdhury, Aniket Didolkar, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. 2020a. Augmenting NLP models using latent feature interpolations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6931–6936. https://doi.org/10.18653/v1/2020.coling-main.611

Jindal, Amit, Narayanan Elavathur Ranganatha, Aniket Didolkar, Arijit Ghosh Chowdhury, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. 2020b. SpeechMix - augmenting deep sound recognition using hidden space interpolations. In *INTERSPEECH*, pages 861–865. https://doi.org/10.21437/Interspeech.2020-3147

Karpukhin, Vladimir, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019*, pages 42–47. https://doi.org/10.18653/v1/D19-5506

Kiperwasser, Eliyahu and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327. https://doi.org/10.1162/tacl_a_00101

Kobayashi, Sosuke. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457. https://doi.org/10.18653/v1/N18-2072

Kolomiyets, Oleksandr, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, Short Papers*, pages 271–276.

Kumar, Varun, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long*

Learning for Spoken Language Systems, pages 18–26.

Ling, Wang, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1520–1530. `https://doi.org/10.18653/v1/D15-1176`

Louvan, Samuel and Bernardo Magnini. 2020. Simple is better! Lightweight data augmentation for low resource slot filling and intent classification. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation*, pages 167–177.

Nguyen, Xuan Phi, Shafiq Joty, Kui Wu, and Ai Ti Aw. 2020. Data diversification: A simple strategy for neural machine translation. In *Advances in Neural Information Processing Systems*, volume 33, pages 10018–10029.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1):10–1162. `https://doi.org/10.1162/0891201053630264`

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A Meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. `https://doi.org/10.3115/v1/D14-1162`

Rosa, Rudolf and David Marecek. 2018. CUNI x-ling: Parsing under-resourced languages in CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 187–196.

Rust, Phillip, Jonas Pfeiffer, Ivan Vulic, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? On the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers)*, pages 3118–3135. `https://doi.org/10.18653/v1/2021.acl-long.243`

Şahin, Gözde Gül. 2016. Verb sense annotation for Turkish PropBank via crowdsourcing. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 496–506. `https://doi.org/10.1007/978-3-319-75477-2_35`

Şahin, Gözde Gül and Esref Adali. 2018. Annotation of semantic roles for the Turkish proposition bank. *Language Resources and Evaluation*, 52(3):673–706. `https://doi.org/10.1007/s10579-017-9390-y`

Şahin, Gözde Gül 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. `https://doi.org/10.18653/v1/D18-1545`

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. `https://doi.org/10.18653/v1/P16-1009`

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. `https://doi.org/10.18653/v1/P16-1162`

Singh, Jasdeep, Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. XLDA: Cross-lingual data augmentation for natural language inference and question answering. *arXiv preprint arXiv:1905.11471*. pages 1–10.

Sulubacak, Umut and Gülşen Eryiğit. 2018. Implementing universal dependency, morphology, and multiword expression annotation standards for Turkish language processing. *Turkish Journal of Electrical Engineering & Computer Sciences*, 26(3):1662–1672. `https://doi.org/10.3906/elk-1706-81`

Sulubacak, Umut, Gülşen Eryiğit, and Tuğba Pamay. 2016. IMST: A revisited Turkish dependency treebank. In *1st International Conference on Turkic Computational Linguistics*, pages 1–6.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for*

Computational Linguistics, ACL 2019, Volume 1: Long Papers, pages 4593–4601, https://doi.org/10.18653/v1/P19-1452

Vaibhav, Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. 2019. Improving robustness of machine translation with synthetic noise. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1916–1920. https://doi.org/10.18653/v1/N19-1190

Vania, Clara, 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 2019*, pages 1105–1116, https://doi.org/10.18653/v1/D19-1102

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

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 344–352.

Wang, William Yang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563. https://doi.org/10.18653/v1/D15-1306

Wang, Xinyi, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: An efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861. https://doi.org/10.18653/v1/D18-1100

Wei, Jason and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. 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 6382–6388. https://doi.org/10.18653/v1/D19-1670

Wieting, John and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088. https://doi.org/10.18653/v1/P17-1190

Wu, Xing, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional BERT contextual augmentation. In *Computational Science - ICCS 2019 - 19th International Conference, Proceedings, Part IV*, pages 84–95. https://doi.org/10.1007/978-3-030-22747-0_7

Yoo, Kang Min, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. *CoRR*, abs/2104.08826:1–11. https://doi.org/10.18653/v1/2021.findings-emnlp.192

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

Zeman, Daniel, Joakim Nivre, Mitchell Abrams, et al. 2020. Universal dependencies 2.6. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Zhang, Hongyi, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412v2*

Zhang, Rongzhi, Yue Yu, and Chao Zhang. 2020. SeqMix: Augmenting Active Sequence Labeling via Sequence Mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8566–8579. https://doi.org/10.18653/v1/2020.emnlp-main.691

Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020*, pages 1–43.

Zhang, Xiang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 649–657.

Zheng, Xiaoqing, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pages 6600–6610. https://doi.org/10.18653/v1/2020.acl-main.590

8. Appendix

Appendix J: SRL Preprocessing

Turkish. We use `Modifier`, `Subject` and `Object` dependency labels as LOI and merge the predicate tokens linked with `MWE` (Multi Word Expression), and `Deriv` (Derivation) while performing augmentation. In order to comply with the requirements of the SRL model, we first merge the inflectional groups of the words that are split via derivational boundaries; and then use character sequences of the full word in the SRL model.

Finnish. This data set uses Universal Dependency (UD) formalism, hence we use the same LOIs as in the original study (Şahin and Steedman 2018) for augmentation. These are namely `nsubj`, `dobj`, `iobj`, `obj`, `obl`, and `nmod` for LOIs and `case`, `fixed`, `flat`, `cop`, and `compound` for multi-word predicates. In addition, to standardize the input format to our SRL model, the custom semantic layer annotation used in Finnish PropBank, has been converted to the same CoNLL-09 format.

Spanish, Catalan. Both data sets use the same dependency annotation scheme: `suj` for subject, `cd` for direct, and `ci` for indirect object relations. Furthermore, we shorten the organization names to abbreviations *(e.g., Confederación_Francesa to CF)* since such long sequences cause memory problems for the SRL model.

Czech. `Sb`, `Obj`, and `Atr` dependency labels are used as LOI and the predicate tokens with `Pred` dependency are merged.

Appendix K: UAS Scores

The UAS scores for dependency parsing experiments are given in Tables A1 and A2.

Appendix L: Result Tables without Colors

We visualize the results presented in Section 5 using no colors. POS Tagging, Dependency Parsing, and Semantic Role Labeling results are given in Tables A3 and A4 accordingly.

Table A1
Dependency parsing UAS scores on original (Org) and augmented data sets, where the results of `uuparser` are given at the top, and `biaffine` parser at the bottom. *language*ˆ denotes that *language* is part of the multilingual BERT.

			Token Level	Character Level					Syntactic		
		Org	SR	CI	CSU	CSW	CD	CA	Crop	Rotate	Nonce
uuparser	be	61.78 ± 0.33	56.79 ± 1.18 ††	63.29 ± 0.39 **	61.64 ± 0.84	62.53 ± 1.24 *	61.00 ± 1.02	61.62 ± 0.59	61.13 ± 0.46	61.84 ± 2.64	62.92 ± 1.39
	bxr	28.14 ± 3.80	29.04 ± 3.79 *	29.46 ± 3.46	30.25 ± 2.79 **	29.93 ± 0.90 **	30.24 ± 2.78 **	31.59 ± 1.97 **	29.80 ± 3.08 **	27.82 ± 3.92	27.07 ± 1.90 ††
	kk	44.96 ± 1.51	45.09 ± 1.80	45.34 ± 0.87	46.61 ± 1.82 **	45.60 ± 1.28 *	45.62 ± 1.51 *	45.20 ± 1.49 *	44.45 ± 1.02	43.01 ± 1.48	44.75 ± 1.62
	ku	32.71 ± 1.10	35.02 ± 1.06 **	35.85 ± 0.22 **	35.98 ± 0.47 **	35.08 ± 1.08 *	37.59 ± 0.50 **	36.54 ± 0.24 **	34.35 ± 0.35 **	38.01 ± 0.54 **	37.49 ± 0.11 **
	ta	64.01 ± 2.02	63.21 ± 2.13	64.04 ± 2.34	63.74 ± 0.86	63.00 ± 1.61	64.62 ± 1.71	63.78 ± 1.12	64.07 ± 1.12	65.11 ± 0.28 *	65.69 ± 0.49 *
	te	87.80 ± 0.67	88.19 ± 0.62	87.75 ± 0.99	88.33 ± 0.98	88.10 ± 0.77	88.66 ± 0.73 *	88.26 ± 0.64	88.06 ± 0.89	88.09 ± 0.94	–
	vi	64.89 ± 0.33	59.10 ± 0.54 ††	62.79 ± 0.49 †	62.79 ± 0.22 †	62.17 ± 0.31 ††	61.87 ± 0.53 ††	61.91 ± 0.81 ††	65.70 ± 0.98	64.84 ± 0.76	–
biaffine	beˆ	82.81 ± 0.45	85.03 ± 0.92 *	85.29 ± 0.39 **	84.69 ± 1.12 *	84.01 ± 0.65	85.07 ± 0.72 **	85.74 ± 0.39 **	84.25 ± 0.75 **	83.72 ± 1.10	83.76 ± 0.45
	bxr	29.67 ± 0.66	31.24 ± 0.42 **	30.81 ± 0.28	30.45 ± 0.33	29.89 ± 0.46	30.03 ± 0.93	30.53 ± 0.68 *	29.33 ± 0.42 †	29.85 ± 0.71	29.16 ± 0.82
	kkˆ	50.85 ± 0.51	52.50 ± 0.67 **	52.80 ± 0.56 **	54.30 ± 0.83 **	54.21 ±0.38 **	53.27 ± 0.67 **	53.29 ± 0.35 **	50.44 ± 0.35	48.92 ± 0.52 †	54.38 ± 0.75 **
	ku	30.48 ± 0.48	30.23 ± 0.15	32.27 ± 0.59 **	30.32 ± 0.25	32.11 ± 0.56 **	32.55 ± 0.46 **	33.46 ± 0.66 **	30.33 ± 0.70	28.78 ± 0.52 ††	33.08 ± 0.37 **
	taˆ	77.39 ± 0.43	79.11 ± 0.47 **	77.81 ± 0.26	78.65 ± 1.18 *	77.86 ± 0.60	78.77 ± 0.63 **	78.85 ± 0.36 **	78.23 ± 0.48 **	78.45 ± 0.91 *	77.88 ± 0.85
	teˆ	91.57 ± 0.57	91.82 ± 0.82	91.35 ± 0.78 †	91.96 ± 0.43	92.09 ± 0.62 *	91.40 ± 0.40	92.23 ± 0.81 *	92.51 ± 0.32 **	91.75 ± 0.21	–
	viˆ	74.06 ± 0.21	69.24 ± 0.79 ††	71.90 ± 0.26 ††	72.23 ± 0.91 ††	72.30 ± 0.87 ††	72.40 ± 0.32 ††	72.61 ± 0.42 ††	73.74 ± 0.33 ††	73.78 ± 0.5	–

Table A2
Dependency parsing UAS scores on original (Org) and augmented data sets, where the results of `uuparser` are given at the top, and `biaffine` parser at the bottom. *language*ˆ denotes that *language* is part of the multilingual BERT. Significance denoted with symbols only.

			Token Level	Character Level					Syntactic		
		Org	SR	CI	CSU	CSW	CD	CA	Crop	Rotate	Nonce
uuparser	be	61.78 ± 0.33	56.79 ± 1.18 ††	63.29 ± 0.39 **	61.64 ± 0.84	62.53 ± 1.24 *	61.00 ± 1.02	61.62 ± 0.59	61.13 ± 0.46	61.84 ± 2.64	62.92 ± 1.39
	bxr	28.14 ± 3.80	29.04 ± 3.79 *	29.46 ± 3.46	30.25 ± 2.79 **	29.93 ± 0.90 **	30.24 ± 2.78 **	31.59 ± 1.97 **	29.80 ± 3.08 **	27.82 ± 3.92	27.07 ± 1.90 ††
	kk	44.96 ± 1.51	45.09 ± 1.80	45.34 ± 0.87	46.61 ± 1.82 **	45.60 ± 1.28 *	45.62 ± 1.51 *	45.20 ± 1.49 *	44.45 ± 1.02	43.01 ± 1.48	44.75 ± 1.62
	ku	32.71 ± 1.10	35.02 ± 1.06 **	35.85 ± 0.22 **	35.98 ± 0.47 **	35.08 ± 1.08 *	37.59 ± 0.50 **	36.54 ± 0.24 **	34.35 ± 0.35 **	38.01 ± 0.54 **	37.49 ± 0.11 **
	ta	64.01 ± 2.02	63.21 ± 2.13	64.04 ± 2.34	63.74 ± 0.86	63.00 ± 1.61	64.62 ± 1.71	63.78 ± 1.12	64.07 ± 1.12	65.11 ± 0.28 *	65.69 ± 0.49 *
	te	87.80 ± 0.67	88.19 ± 0.62	87.75 ± 0.99	88.33 ± 0.98	88.10 ± 0.77	88.66 ± 0.73 *	88.26 ± 0.64	88.06 ± 0.89	88.09 ± 0.94	–
	vi	64.89 ± 0.33	59.10 ± 0.54 ††	62.79 ± 0.49 †	62.79 ± 0.22 †	62.17 ± 0.31 ††	61.87 ± 0.53 ††	61.91 ± 0.81 ††	65.70 ± 0.98	64.84 ± 0.76	–
biaffine	beˆ	82.81 ± 0.45	85.03 ± 0.92 *	85.29 ± 0.39 **	84.69 ± 1.12 *	84.01 ± 0.65	85.07 ± 0.72 **	85.74 ± 0.39 **	84.25 ± 0.75 **	83.72 ± 1.10	83.76 ± 0.45
	bxr	29.67 ± 0.66	31.24 ± 0.42 **	30.81 ± 0.28	30.45 ± 0.33	29.89 ± 0.46	30.03 ± 0.93	30.53 ± 0.68 *	29.33 ± 0.42 †	29.85 ± 0.71	29.16 ± 0.82
	kkˆ	50.85 ± 0.51	52.50 ± 0.67 **	52.80 ± 0.56 **	54.30 ± 0.83 **	54.21 ±0.38 **	53.27 ± 0.67 **	53.29 ± 0.35 **	50.44 ± 0.35	48.92 ± 0.52 †	54.38 ± 0.75 **
	ku	30.48 ± 0.48	30.23 ± 0.15	32.27 ± 0.59 **	30.32 ± 0.25	32.11 ± 0.56 **	32.55 ± 0.46 **	33.46 ± 0.66 **	30.33 ± 0.70	28.78 ± 0.52 ††	33.08 ± 0.37 **
	taˆ	77.39 ± 0.43	79.11 ± 0.47 **	77.81 ± 0.26	78.65 ± 1.18 *	77.86 ± 0.60	78.77 ± 0.63 **	78.85 ± 0.36 **	78.23 ± 0.48 **	78.45 ± 0.91 *	77.88 ± 0.85
	teˆ	91.57 ± 0.57	91.82 ± 0.82	91.35 ± 0.78 †	91.96 ± 0.43	92.09 ± 0.62 *	91.40 ± 0.40	92.23 ± 0.81 *	92.51 ± 0.32 **	91.75 ± 0.21	–
	viˆ	74.06 ± 0.21	69.24 ± 0.79 ††	71.90 ± 0.26 ††	72.23 ± 0.91 ††	72.30 ± 0.87 ††	72.40 ± 0.32 ††	72.61 ± 0.42 ††	73.74 ± 0.33 ††	73.78 ± 0.5	–

Table A3
Dependency parsing LAS scores on original (Org) and augmented data sets, where the results of `uuparser` are given at the top, and `biaffine` parser at the bottom. *language*ˆ denotes that *language* is part of the multilingual BERT. Significance denoted with symbols only.

			Token Level	Character Level					Syntactic		
		Org	SR	CI	CSU	CSW	CD	CA	Crop	Rotate	Nonce
uuparser	be	52.85 ± 0.70	50.10 ± 0.67 ††	54.05 ± 0.40 **	53.39 ± 0.51	55.37 ± 1.48 *	54.89 ± 0.87 *	53.57 ± 0.97 *	53.01 ± 1.01	53.03 ± 0.31	54.27 ± 0.68
	bxr	11.73 ± 1.65	12.70 ± 1.84 *	12.39 ± 1.31	13.04 ± 1.20 **	13.83 ± 0.46 **	13.44 ± 1.06 **	14.06 ± 1.14 **	11.99 ± 1.52 **	11.43 ± 1.39	10.70 ± 0.77 ††
	kk	23.82 ± 0.92	23.82 ± 0.98	24.36 ± 0.59	24.80 ± 0.75 **	24.87 ± 0.77 *	24.79 ± 0.86 *	24.62 ± 0.83 *	24.09 ± 0.79	23.76 ± 0.32	24.78 ± 1.07 *
	ku	21.09 ± 0.86	23.22 ± 0.60 **	23.70 ± 0.92 **	24.08 ± 0.74 **	23.48 ± 0.87 *	24.32 ± 0.55 **	24.39 ± 0.12 **	22.74 ± 0.35 **	24.50 ± 0.44 **	23.98 ± 0.31 **
	ta	55.52 ± 0.39	54.27 ± 1.99	55.39 ± 0.42	55.15 ± 0.71	54.89 ± 0.90	56.11 ± 0.77	56.52 ± 1.31	54.62 ± 0.65 *	55.16 ± 0.18 *	57.50 ± 0.43 **
	te	77.79 ± 0.85	76.94 ± 0.94	77.65 ± 0.94	78.83 ± 0.73 *	77.27 ± 1.12 *	78.11 ± 0.80 **	78.87 ± 0.51	78.01 ± 0.86 *	78.27 ± 1.05 *	–
	vi	55.01 ± 0.15	48.23 ± 0.27 ††	53.80 ± 0.32	52.26 ± 0.47 †	52.19 ± 0.36 ††	52.92 ± 0.23 ††	53.15 ± 0.42 †	55.33 ± 0.40	54.87 ± 0.26	–
biaffine	beˆ	78.17 ± 0.36	79.66 ± 0.28	80.47 ± 0.11 **	80.04 ± 0.80 **	78.79 ± 0.64	80.58 ± 0.49 **	80.45 ± 0.36 **	79.84 ± 0.30 **	79.78 ± 0.35 **	79.33 ± 0.27 **
	bxr	17.71 ± 0.52	18.73 ± 0.13 **	17.91 ± 0.51	17.65 ± 0.44	17.73 ± 0.17	17.72 ± 0.87	17.81 ± 0.50 *	16.74 ± 0.88 †	17.59 ± 0.36	17.12 ± 0.82
	kkˆ	34.23 ± 0.32	35.49 ± 0.75 **	35.61 ± 0.52 **	36.69 ± 0.76 **	36.30 ±0.59 **	35.42 ± 0.49 **	35.76 ± 0.42 **	34.61 ± 0.75 *	34.18 ± 0.42	38.43 ± 0.17 **
	ku	20.48 ± 0.35	20.64 ± 0.24	22.16 ± 0.47 **	20.49 ± 0.33	22.56 ± 0.68 **	22.71 ± 0.30 **	23.13 ± 0.26 **	20.53 ± 0.41	19.39 ± 0.50 ††	22.82 ± 0.24 **
	taˆ	70.01 ± 0.62	70.99 ± 0.21 **	70.27 ± 0.10	71.21 ± 1.08 *	70.54 ± 0.49	71.13 ± 0.67 **	71.47 ± 0.53 **	70.89 ± 0.30 **	71.11 ± 1.01 *	71.24 ± 0.85 **
	teˆ	84.60 ± 0.79	84.60 ± 0.80	84.28 ± 0.97 †	84.88 ± 0.54 **	85.02 ± 0.32 **	84.05 ± 0.78 ††	84.74 ± 0.81	85.30 ± 0.57 **	84.19 ± 0.59 ††	–
	viˆ	66.25 ± 0.84	62.40 ± 0.76 ††	63.49 ± 0.80 ††	63.87 ± 0.55 ††	64.22 ± 2.07 ††	64.54 ± 2.05 ††	64.09 ± 1.25 ††	66.11 ± 0.85	65.94 ± 0.91	–

Table A4
Semantic role labeling results on original (Org) and augmented data sets. Significance denoted with symbols only.

Catalan									
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	31.34 ± 0.99	37.29 ± 1.26 **	38.50 ± 1.37 **	37.74 ± 1.64 **	39.79 ± 0.25 **	37.45 ± 1.58 **	38.28 ± 1.48 **	26.59 ± 0.52 ††	25.96 ± 0.62 ††
500	44.53 ± 1.39	44.12 ± 1.78	49.75 ± 0.10 **	47.77 ± 0.65 *	46.81 ± 1.59 *	47.47 ± 0.62 *	49.46 ± 0.19 **	44.30 ± 1.16 †	44.72 ± 0.36
1,000	53.06 ± 1.33	53.49 ± 0.64	53.80 ± 0.67	54.83 ± 0.81	56.37 ± 0.29 *	55.50 ± 0.17 *	53.97 ± 0.96	53.57 ± 0.07	52.31 ± 0.96
Turkish									
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	31.28 ± 0.12	31.78 ± 1.20	31.62 ± 1.53	30.67 ± 2.19	30.14 ± 2.46	32.52 ± 0.89 *	32.02 ± 1.53	32.42 ± 0.64 *	30.37 ± 1.75
500	35.75 ± 1.38	35.95 ± 0.65	36.35 ± 0.68	38.27 ± 0.88 *	37.30 ± 1.05	34.80 ± 1.96	36.23 ± 1.37	37.40 ± 1.07 *	34.58 ± 1.89
1,000	44.89 ± 0.80	42.41 ± 1.20	43.36 ± 0.67	41.78 ± 1.39	42.28 ± 1.31	41.42 ± 1.38	29.14 ± 0.47 ††	44.83 ± 1.07	42.71 ± 0.76
Spanish									
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	31.23 ± 1.30	34.65 ± 1.02 *	36.31 ± 2.28	37.91 ± 1.21 **	36.80 ± 1.90 *	36.76 ± 2.11 *	37.34 ± 1.14 **	26.04 ± 1.34 ††	26.95 ± 2.22 ††
500	44.16 ± 0.68	43.89 ± 0.65	44.80 ± 0.81	44.82 ± 1.10	43.75 ± 1.36	44.60 ± 0.72 *	45.03 ± 0.91 *	42.35 ± 0.42 ††	41.13 ± 0.38 ††
1,000	50.98 ± 1.30	50.80 ± 1.21	53.03 ± 0.76 **	52.53 ± 0.60 **	52.10 ± 1.00 *	52.76 ± 0.57 **	54.12 ± 0.12 **	51.32 ± 1.09	52.40 ± 0.34 *
Czech									
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	35.63 ± 1.08	33.48 ± 1.58	37.17 ± 0.13 *	37.56 ± 0.02 *	31.56 ± 1.00 †	35.70 ± 0.80	36.14 ± 1.78	34.84 ± 1.55 **	30.78 ± 0.38 ††
500	44.96 ± 0.51	42.04 ± 1.08 †	46.44 ± 1.20 *	46.13 ± 1.97	46.24 ± 1.60	47.92 ± 0.59 **	47.42 ± 1.07 *	46.38 ± 0.12 *	44.83 ± 1.47
1,000	50.29 ± 0.09	48.15 ± 0.31 †	48.66 ± 0.49 †	52.63 ± 0.97 *	51.10 ± 1.23	47.85 ± 1.19 †	51.97 ± 0.54 *	49.33 ± 1.36	48.45 ± 1.34
Finnish									
#sample	**Org**	**SR**	**CI**	**CSU**	**CSW**	**CD**	**CA**	**Crop**	**Rotate**
250	18.80 ± 0.89	18.60 ± 2.34	17.18 ± 1.56	19.68 ± 1.16	25.71 ± 0.80 **	28.87 ± 1.55 **	27.96 ± 0.44 **	30.83 ± 0.17 **	20.07 ± 2.42
500	37.23 ± 0.35	34.10 ± 1.61	35.59 ± 0.87 †	36.88 ± 0.29 †	38.98 ± 0.03 *	35.29 ± 0.99 †	35.32 ± 1.63	37.58 ± 0.93 *	35.16 ± 1.08
1,000	41.64 ± 0.98	40.15 ± 0.95 †	41.27 ± 0.84	41.41 ± 1.08	39.80 ± 0.81	40.57 ± 1.17	41.99 ± 0.04	41.35 ± 1.37	39.08 ± 0.34 †