Towards Incremental NER Data Augmentation via Syntactic-aware Insertion Transformer

Wenjun Ke[1] , **Zongkai Tian**[2] , **Qi Liu**[2] , **Peng Wang**[1*] , **Jinhua Gao**[3] , **Rui Qi**[4]
[1]School of Computer Science and Engineering, Southeast University
[2]Beijing Institute of Computer Technology and Application
[3]Institute of Computing Technology, Chinese Academy of Sciences
[4]China Life Property Casualty Insurance Company Limited
kewenjun@ict.ac.cn, tzk_bit@163.com, liuqi970811@163.com, pwang@seu.edu.cn,
gaojinhua@ict.ac.cn, qirui0817@gmail.com

Abstract

Named entity recognition (NER) aims to locate and classify named entities in natural language texts. Most existing high-performance NER models employ a supervised paradigm, which requires a large quantity of high-quality annotated data during training. In order to help NER models perform well in few-shot scenarios, data augmentation approaches attempt to build extra data by means of random editing or by using end-to-end generation with PLMs. However, these methods focus on only the fluency of generated sentences, ignoring the syntactic correlation between the new and raw sentences. Such uncorrelation also brings low diversity and inconsistent labeling of synthetic samples. To fill this gap, we present SAINT (**S**yntactic-**A**ware **I**nsertio**N** **T**ransformer), a hard-constraint controlled text generation model that incorporates syntactic information. The proposed method operates by inserting new tokens between existing entities in a parallel manner. During insertion procedure, new tokens will be added taking both semantic and syntactic factors into account. Hence the resulting sentence can retain the syntactic correctness with respect to the raw data. Experimental results on two benchmark datasets, i.e., Ontonotes and Wikiann, demonstrate the comparable performance of SAINT over the state-of-the-art baselines.

1 Introduction

Named entity recognition (NER), as an important fundamental technique in information extraction, is of great significance to many downstream tasks such as text summarization [Nallapati *et al.*, 2016] and information retrieval [Banerjee *et al.*, 2019]. Methods utilizing pre-trained language models (PLMs) have achieved outstanding performance [Peters *et al.*, 2017; Souza *et al.*, 2020]. However, such techniques primarily follow a supervised paradigm, and high performance primarily hinges on the size and quality of the training data. Actually, the major challenge of this technique lies in the manual annotation of the corpus, which requires domain knowledge and is time-consuming, ending in the absence of annotated data over various verticals.

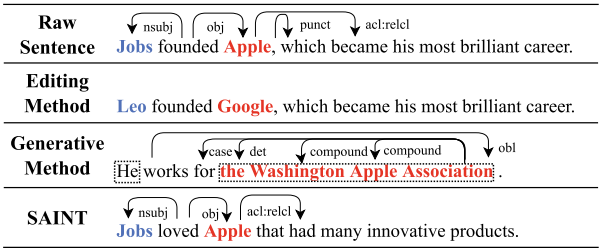


Figure 1: An illustration of the editing and generative methods, where different colors denote different entity types and the dotted boxes represent the location of errors.

Several methods have been proposed to alleviate the above challenge, such as meta-learning [Wu *et al.*, 2020; de Lichy *et al.*, 2021; Ma *et al.*, 2022c], prompt learning [Ma *et al.*, 2022b; Liu *et al.*, 2022a; Chen *et al.*, 2022], and data augmentation [KERAGHEL *et al.*, 2020; Phan and Nguyen, 2022]. Among these, explicit data augmentation could be easily and seamlessly plugged into other few-shot NER approaches. Generally, the primary technical methods of NER data augmentation research can be grouped into two paradigms: editing [Zhang *et al.*, 2015; Cai *et al.*, 2020; Wei and Zou, 2019; Min *et al.*, 2020] and generative [Sun and He, 2020; Yoo *et al.*, 2019; Yu *et al.*, 2018; Zhou *et al.*, 2022] methods. Editing techniques create new data by modifying the raw text with word substitution [Zhang *et al.*, 2015; Cai *et al.*, 2020], random deletion [Wei and Zou, 2019], and order exchange [Min *et al.*, 2020]. In this vein, the editing method might raise issues such as the diversity deficiency of synthetic samples, thus reducing the generalization power of NER models. For example, in the second case of Figure 1, when *Jobs* and *Apple* were substituted with *Leo* and *Google* respectively, the revised sentence was basically unchanged with the same context as the raw sentence.

In comparison with editing methods, the generative methods can generate samples with higher fluency and variety e.g., GANs [Sun and He, 2020], VAEs [Yoo *et al.*, 2019], translation [Yu *et al.*, 2018] and prompt learning models [Wang *et*

*Corresponding Author

al., 2022; Liu *et al.*, 2022b]. Nevertheless, the performance of existing generative methods may suffer from two disadvantages.On the one hand, compared with the raw sentence, the resulting sequence may cause syntactic deviation. As a result, the boundary of the generated entity may change, and the original entity label may not correspond correctly to the generated entity. Consider the second case in Figure 1, where the entity *Apple* was incorrectly expanded into another entity words *the Washington Apple Association* with a wider entity boundary. On the other hand, the generative method with a prompt learning strategy encourages the assembly of entities to guide sentence generation but lacks specific hard constraints to ensure entity occurrence. However, we are unable to obtain the ground truth labels of the newly generated token in advance. As shown in the previous example, the entity *Jobs* was dismissed during generation and replaced by a non-entity word *He*.

To address these issues, we argue that the key of the NER data augmentation task lies in preserving the semantic diversity and syntactic consistency between created sentences and raw ones. In this paper, we propose SAINT (**S**yntactic-**A**ware **I**nsertio**N** **T**ransformer), a hard-constrained controlled text generative model that explicitly incorporates the syntactic feature into pre-trained language models (PLMs) and restricts the appearance of predefined entities in the generated sentence. In contrast to previous generative approaches, SAINT generates sentences while maintaining semantic diversity and syntactic relevance with respect to the raw data. Specifically, we first turn the raw sentence into a dependency tree and obtain its flattened form containing the dependency direction and distance. Inspired by a hard constraint text generation work [Zhang *et al.*, 2020], the training dataset is constructed by randomly picking and masking key tokens and corresponding syntactic annotations in raw sentences. The processed outputs are then fed into SAINT with the flattened dependency structures and the token themselves as the supervised objectives for multi-task training. In the inference stage, we feed the predefined entities and keywords into SAINT, generating the context words and corresponding syntactic annotations incrementally. The final training sentence can be selected using a sophisticated dependency similarity comparison algorithm.

Experimental results on two benchmark datasets from Wikiann [Pan *et al.*, 2017] and OntoNotes [Hovy *et al.*, 2006] demonstrate the superior performance of our model over state-of-the-art NER data augmentation baselines, with improvements of 2.21% and 3.13% over the strongest baseline model in the best case, respectively. The effectiveness of introducing syntactic features for the low-resource NER data augmentation task is also verified. Moreover, further analysis reveals the ability of SAINT to retain the dependency consistency of the raw sentence through the sample generation process. To sum up, the major contributions of this paper are three-folds:

- We argue that ensuring semantic diversity and syntactic consistency with the raw data is crucial for NER data augmentation. Based on this consideration, we construct the training dataset with both tokens and corresponding syntactic labels under the few shot settings.
- We propose a novel NER data augmentation model, SAINT, which explicitly incorporates syntactic features into the insertion transformer to generate samples containing preset entities parallelly.
- Extensive experiments are conducted on the Wikiann and OntoNotes datasets to verify the effectiveness of our method and demonstrate the significance of introducing syntactic features.

2 Method

Given a sentence x_i from the downstream NER corpus $X = \{x_0, x_1, ..., x_n\}$, the ultimate goal of SAINT is to produce a set of new sentences $y_i = \{y_i^0, y_i^1, ..., y_i^m\}$ for each input x_i, where n denotes the size of original corpus, and m is the number of generation scale. The generative data y_i should be grammatically correct and semantically consistent with the raw sentence x_i. The total of $y_i\,(0 \leq i \leq n)$ together makes up the generated dataset $Y = \{y_0, y_1, ..., y_n\}$ that is further used in augmentation.

The overall architecture of SAINT is shown in Figure 2. In the first stage, SAINT constructs the training samples s_i for each data x_i by utilizing the flattened form of the dependency tree for label syntactic features and picking and masking tokens dynamically. In the second stage, the syntactic feature is injected into the pre-trained language model, and the hard constraint text generation paradigm restricts the appearance of predefined entities in the generated sentence. The multi-task learning method could supervise the model to predict tokens and corresponding dependency labels. In the last stage, SAINT incrementally generates the sentence in a token-by-token manner. Moreover, the syntactic similarity measurement helps select the most syntactic-relevant sentence as the final result.

2.1 Training Sample Creation

The first stage of our method is to create samples for training. Due to the difficulty of directly constructing a stepwise mask loss function, the model is trained with data instance pairs. The training data is organized into a sequence of $(raw_{token}, raw_{syntax}, predict_{token}, predict_{syntax})$ pairs after progressive masking. Specifically, for each sample in the raw dataset, its syntactic dependency labels can be obtained by syntactic feature labeling, and tokens are masked from stage k to stage $k+1$ according to the importance score by the token selector. Consequencely, every interval pair $(s_i^k,\ s_i^{k+1})$ comprises the training data. For example, sample {(studies, Park, College), (-0obj,+0compound,-0nmod)} and sample {(continued, studies, Park, College), (-0ROOT,-0obj,+0compound,-0nmod)} form a data pair, as illustrated in Figure 2.

Syntactic Feature Labeling

As a syntactic dependency contains the information of the direction, distance, and relation type, we define syntactic dependency labels as follows:

$$(+|-)(\ell)(dep) \tag{1}$$

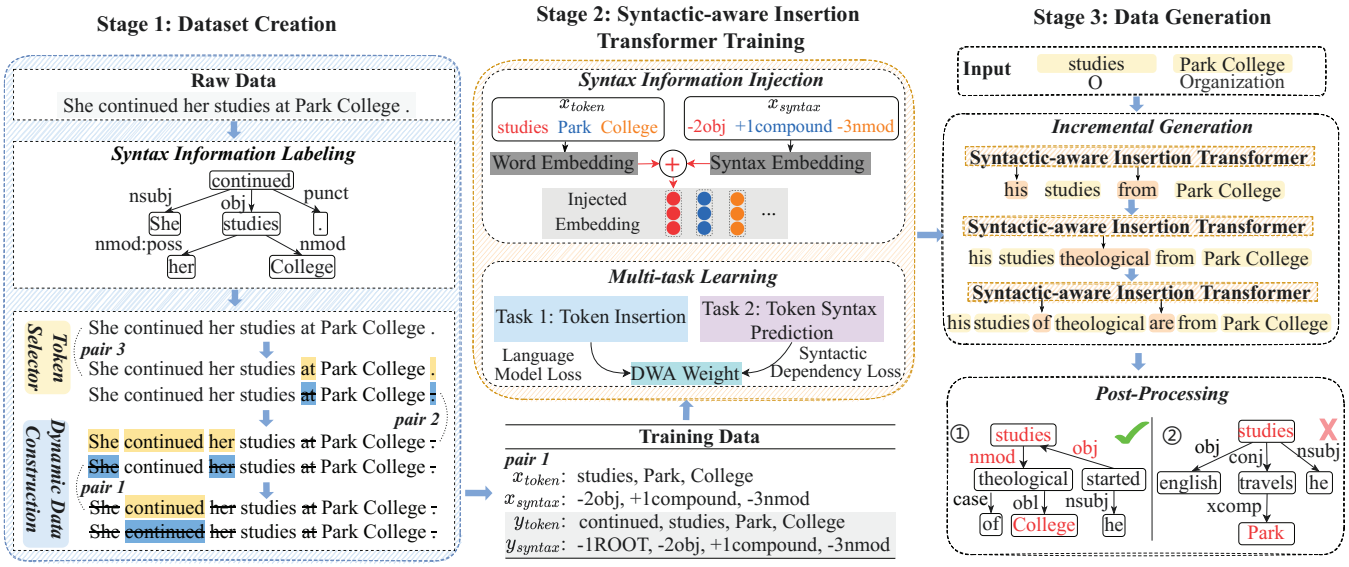


Figure 2: Overview of SAINT's workflow with respect to the sentence *She continued her studies at Park College.*

where $(+|-)$, ℓ, and dep denote such three parts respectively. $(+|-)$ designates the direction between the head and the dependent, where + denotes the position of the head in the sentence as coming before the dependent, and - is the opposite. ℓ indicates the distance between two tokens. Due to the extensive range of distance, it may result in a large number of labels. To avoid the label sparsity problem in the low-resource scenarios, we further process the value of distance to ℓ. We assign the value of ℓ to 0 when the distance is less than 5, and to 1 when the distance is equal to or greater than 5. dep shows the dependency relation type that the head belongs to.

To get the defined syntactic labels, we first apply the Stanford Parser[1] to obtain the syntactic dependencies of each sample. For the triplets (tail, type, head) in syntactic dependencies, we use the absolute value of relation *head* to relation *tail* as the distance and connect them to form the relation labels. For the case where a word corresponds to multiple labels, we only retain the labels of the predefined important syntactic component. Secondly, the formed label is then matched to the original word segmentation result to form syntactic dependency labels. Finally, we can obtain the syntactic dependency labels with respect to each sample. As shown in Figure 2, under the above process, the sample {She continued her studies ...} can be reflect to its corresponding syntactic label {+0nsubj, -0ROOT, +0nmod:pass, -0obj ...}.

Token Selector

To assess the significance of the tokens in the sentence, we consider the four different token scoring schemes. The score indicates the importance of tokens for different stage masks. Concretely, the importance score of the token x_i in stage X^k is defined as:

$$p_i = p_i^{tf-idf} + p_i^{pos} + p_i^{yake} + p_i^{dep} \tag{2}$$

where p_i^{tf-idf}, p_i^{pos}, p_i^{yake}, and p_i^{dep} represent the scores of TF-IDF, POS tagging, YAKE [Campos *et al.*, 2020], and syntactic feature, respectively, and each item is adjusted to range 0 to 1. We increase the score of important syntactic components to highlight their contribution and keep the scores of other syntactic components unchanged. Throughout the construction of the dataset, the less important tokens are selected at an earlier stage, and then feed into the dynamic data construction module to determine whether to mask.

Dynamic Data Construction

Since SAINT does not insert multiple tokens between two adjacent tokens under the step-by-step data generation process, there should be no continuous masking of two tokens during the construction of training data. Based on such consideration, we calculate p_i of each token in each step and choose a set of tokens with the highest total score. Formally, this problem can be defined as:

$$\begin{aligned} &\max \sum_{i}^{L} \alpha_i (p_{\max} - p_i) \\ &s.t. \quad \alpha_i \alpha_{i+1} \neq 1 \quad \forall i \end{aligned} \tag{3}$$

Suppose the current sequence is $s_i^k = \{w_0, w_1, \cdots, w_n\}$, where α_i denotes whether w_i is masked or not. $\alpha_i \in \{0, 1\}$, $\alpha_i = 0$ signifies keeping the corresponding token w_i, while $\alpha_i = 1$ implies masking w_i. $p_{\max} = \max\{p_i\}$. Using $p_{\max}$ to subtract the score of each token to ensure that all the scores calculated by the algorithm are positive. Then we can conduct a dynamic programming [Gries, 1982; Zhang *et al.*, 2020] to select tokens that should be masked.

2.2 Syntactic-aware Insertion Transformer Training

In this section, we introduce the training process of syntactic-aware insertion transformer using the training dataset obtained in the previous phrase. Specifically, raw_{token} is fed into the model, while syntactic feature raw_{syntax} is also injected simultaneously. Moreover, the model is supervised to predict token $predict_{token}$ and corresponding dependency labels $predict_{syntax}$.

[1] https://stanfordnlp.github.io/CoreNLP/

Syntactic Feature Injection

In the training stage, we combine both embeddings of the syntactic dependency labels raw_{syntax} and word raw_{token}. As shown in Equation (4) :

$$\mathbf{e}_i = \lambda_s \mathbf{e}_i^{word} + (1 - \lambda_s)\mathbf{e}_i^{syntax} \tag{4}$$

where $\mathbf{e}_i^{word} \in \mathbb{R}^{d_w}$ and $\mathbf{e}_i^{syntax} \in \mathbb{R}^{d_w}$ are low-dimensional and dense vectors initialized by word embedding and syntax embedding, respectively. $\mathbf{e}_i \in \mathbb{R}^{d_w}$ serves as the representation of token w_i. $d_w = 768$ denotes the dimension of the word vector. λ_s is a hyper parameter manually specified to balance the weight of syntax and semantic.

Multi-task Learning

Given all the input sample pairs (s_i^k, s_i^{k+1}) for the k^{th} pair of i^{th} sample, Our training process consists of two tasks: pre-trained language model fine-tuning and syntactic dependency label prediction. The first item refers to calculating the loss of the language model $\mathcal{L}_{\ell m}$ as:

$$\mathcal{L}_{\ell m} = -\log p(s_i^{k+1}|s_i^k) \tag{5}$$

Moreover, $\mathcal{L}_{dep}$ indicates the cross-entropy loss towards the syntactic dependency labels, which can be calculated by Equation (6) :

$$\mathcal{L}_{dep} = -\sum_{v_i \in V} v_i \log \widetilde{v_i} \tag{6}$$

where v_i and $\widetilde{v_i}$ denote the ground truth and predicted syntactic dependency labels, respectively.

In the training phase, teacher-forcing and multi-task learning strategies are employed. And the Dynamic Weight Averaging (DWA) [Liu *et al.*, 2019] is utilized to dynamically balance the two tasks and obtain weights of the PLM fine-tuning task and the syntactic dependency prediction task as $\lambda_{\ell m}$ and λ_{dep}. In summary, the total loss of the syntactic-aware insertion transformer can be computed as:

$$\mathcal{L} = \lambda_{\ell m} \cdot \mathcal{L}_{\ell m} + \lambda_{dep} \cdot \mathcal{L}_{dep} \tag{7}$$

where $\lambda_{\ell m}$ and λ_{dep} are both in the range 0 to 1, which are allocated automatically by DWA mechanism.

2.3 Data Generation

The data generation process started with the given keyword sequence raw_{token}, and the trained SAINT is used to generate candidate insertion words incrementally. In addition, with a large amount of generated data, we use Bayesian risk minimization to select the sample that has similar syntactic features to the raw sentence.

Incremental Generation

Since each step of the generation process generates tokens simultaneously, tokens inserted into different slots may be unaware of others, resulting in inconsistent expressions or duplicate content. We employ a beam search algorithm with Minimum Bayes Risk (MBR) [Kumar and Byrne, 2004] to select label sequence alternatives at each time step, limiting the number of alternatives to the beam width B.

Suppose the existing token sequence as $s^k = \{s_1^k, s_2^k, \cdots, s_{l_k}^k\}$ and the sequence length as l_k. By executing an insertion procedure, it will predict the top B candidate tags in each slot, and the generated candidate sequence is $C_k = \{c_1^{(b)}, c_2^{(b)}, \cdots, c_i^{(b)}\}$, where $c_j^{(b)}, j \subset \{1, \cdots, i\}, b \subset \{1, \cdots, B\}$ is one of the top B candidate tokens in the current j^{th} slot. During the insertion of candidate words in the next slot, the model generates the most likely top B candidate token sequences. By iteratively executing the above steps, the top B generated sequences with the largest likelihood are reserved as the final candidate set S.

Dependency Similarity Comparison

For the top B generated candidate sentences, some of the generated syntactic labels may not align with the actual labels. We argue that generated sentences can be more in line with raw data when the generated syntactic labels are highly consistent with the real ones. To this end, we perform a dependency similarity comparison method, and select the most syntactic-relevant result. Particularly, for each sentence s_i in the candidate set S, the generated syntactic label sequence is $gs_i = \{ge_1, ge_2, \cdots, ge_{im}\}$, the actual syntactic label sequence obtained by the Stanford Parser is $as_i = \{ae_1, ae_2, \cdots, ae_{im}\}$. The dependency similarity score of sentence s_i can be obtained as:

$$score_i = \frac{1}{\mathrm{M}} \sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{sim}\left(ge_{im}, ae_{im}\right) \tag{8}$$

where the function $\mathrm{sim}(\, ge_{im}, ae_{im}\,)$ denotes to calculate the similarity between ge_{im} and ae_{im} with fuzzy comparison. $score_i = 1$ if the syntactic distance between labels is less than 2 and they have the same dependency relation type, and $score_i = 0$ otherwise. Finally, we select the sentence s_{best} with the highest dependency similarity score in the candidate set S as the final generation result.

3 Experiments

In this section, we introduce the experimental settings and baseline methods and report the results of the experiments.

3.1 Datasets and Experimental Settings

We conduct experiments on the two NER datasets of Wikiann [Pan *et al.*, 2017] and OntoNotes [Hovy *et al.*, 2006]. The Wikiann dataset includes three types of entities: location, organization, and person. The OntoNotes dataset contains 18 types of entities, e.g., cardinal, date, event, law, and language. In the experiment, we sampled 100, 200, 300, 400, and 500 items of data as training data under low resource constraints.

Our method is implemented with Python 3.7.12, PyTorch 1.8.0, and the NVIDIA Tesla V100 16G platform. In the experiment, SAINT utilized the Bert-base model as the pre-trained language model and adopted the AdamW optimizer with a learning rate of 1e-5. The training batch size is 20, and the model is trained for 5 epochs. The inference process is performed in three steps. The parameter λ_s is adjusted to 0.2. The NER model's optimizer in the experiment is the AdamW optimizer, the learning rate is 2e-5, and the batch

size is 8. The model with the best validation set performance in 20 epochs is chosen as the final model for testing.

In order to fairly evaluate the performance of each method, we use the XML-RoBERTa-Large [Conneau *et al.*, 2020] with CRF [Lample *et al.*, 2016] as the unified NER model. Moreover, we test three times and metric the average Micro-F1 value as the final result.

3.2 Baselines

We choose seven representative editing and generative NER data augmentation approaches as baselines for comparison.

Editing Methods

Label-wise token replacement (LwTR) leverages the binomial distribution to replace tokens with the same label in the training set. **Synonym replacement (SR)** uses the same token selection process as LwTR and replaces the token with its synonyms from WordNet. **Mention replacement (MR)** replaces entities by sampling the entities with the same labels in the training set. **Shuffle within segments (SiS)** cuts the sentence into several segments and replaces the segment with the same label.

Generative Methods

DAGA [Ding *et al.*, 2020] formulates data augmentation as the sequence tagging task, where the labeled sentences are first linearized. A language model (LM) is then fine-tuned using such linearized data to generate synthetic samples. **MulDA** [Liu *et al.*, 2021] extends [Ding *et al.*, 2020] to low-resource cross-lingual NER, and introduces the process of filtering out the low-quality data. **PromDA** [Wang *et al.*, 2022] first trains the soft prompt using the frozen PLM, then generates synthetic data by deleting useless samples with the help of NLU models.

3.3 Main Results

The experimental results on the two datasets are shown in Table 1, where **no augmentation** implies merely splicing all preset entities together without any other operations.

First, the editing method can typically outperform the non-augmentation manner, with the F1 value of SiS on the 200-sample Wikiann dataset achieving an absolute gain of 4.83%. Furthermore, as the number of samples rises, the performance improvement of these methods gradually slows down. Notably, MR is inferior to no augmentation under the 300, 400, and 500 sample settings on Wikiann. This disadvantage might be due to the low diversity of contexts within the samples obtained through a straightforward replacement operation.

Secondly, comparing the generative and editing methods reveals that the former maintains an absolute lead. An obvious observation is that regarding the 500 samples in the OntoNotes dataset, DAGA, MulDA, PromDA, and our method SAINT increase the F1 value by 9.21%, 10.49%, 12.39% and 14.58% over the best editing method SR, respectively. Besides, PromDA employs the natural language generation (NLG) technique rather than natural language understanding (NLU), achieving promising results on both datasets thanks to the prompt learning strategy.

Finally, our approach SAINT performs better than the other baseline models in most cases. Regarding the 500 samples of the OntoNote dataset, our approach outperforms no augmentation and PromDA in terms of F1 values by 17.98% and 2.19%, respectively. Regarding the 500 samples of the Wikiann dataset, such improvements are 14.23% and 3.13%. Even though SAINT is slightly inferior to PromDA on 100 and 200 samples of the OntoNote dataset, comparable results are also achieved. Note that our method can be better adapted to generate more new samples. On the one hand, it demonstrates the strong generalization ability of our method. On the other hand, such abilities can help resolve complicated low-resource scenarios in reality.

3.4 Ablation Study

The motivation for our approach is to preserve the semantics within the raw sample while ensuring the label correctness of the generated data with the help of syntactic features. To verify the effectiveness of our method, we conduct ablation experiments and show the results in Table 2.

As can be seen, the performance of SAINT on both data sets is severely affected by the removal of the syntactic injection module, with the maximum decline of 3.99% and 4.26%, respectively. Further observation reveals that, with respect to the multi-task learning scheme, the F1 score of SAINT decreases maximally by 2.97% and 4.34% when omitting the language model loss and syntactic loss, respectively. By contrast, the syntactic loss serves as a more significant effect on model performance. Incidentally, analyzing the difference in results between the two datasets, the removal operations appear to have a greater impact on Wikiann than OntoNotes. The major reason is that sentences of Wikiann are more concise and grammatically correct, encouraging syntactic consistency is more critical.

3.5 Effect of Hyper Parameters

In this section, we conduct experiments with different hyper parameters to investigate the effectiveness of syntactic feature injection and multi-task learning. Each experiment utilizes 100 to 500 samples from the training set, with parameters ranging from 0.1 to 0.9. The result is shown in Figure 3.

Effect of Syntactic Feature Injection

We evaluate the influence of syntactic information injection by adjusting the value of λ_s controlling the weight of word embedding and syntax embedding in Equation (4), where the smaller λ_s denotes the more weight of syntax embedding. Figure 3(a) shows that under all datasets and sample number settings, the smaller λ_s (0.2-0.4) presents the optimal result, indicating that syntactical information is more critical than semantic information. More broadly, the best results can be obtained by injecting syntax and semantics simultaneously.

Effect of Multi-task Learning Losses

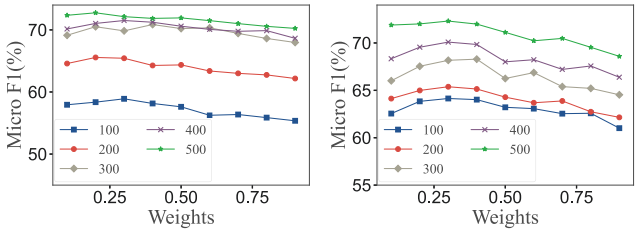
In order to verify the impact of losses on the experimental results, we adjust the weight $\lambda_{\ell m}$ of language model loss and set $\lambda_{dep} = 1 - \lambda_{\ell m}$. Figure 3(b) demonstrates that the Micro F1 value increases noticeably at first, then starts to decline. SAINT performs best when the weights of the two losses are nearly equal. Actually, SAINT concentrates on the

	Model	OntoNotes					Wikiann				
		100	200	300	400	500	100	200	300	400	500
	no augmentation	43.80	49.32	51.19	53.59	54.77	47.19	48.36	51.88	55.01	58.08
editing methods	LwTR	45.22	50.90	53.74	55.62	58.07	48.55	52.51	54.92	56.86	58.61
	SR	45.68	51.94	52.29	55.56	58.17	49.81	53.19	55.17	55.84	57.95
	MR	46.14	51.08	52.25	55.03	55.33	46.20	50.47	50.69	54.26	55.39
	SiS	44.93	49.62	51.85	55.09	56.81	47.21	52.30	54.22	57.28	57.10
generative methods	DAGA	48.78	57.10	59.53	62.05	63.98	50.42	55.63	57.32	58.73	62.16
	MulDA	56.47	61.03	63.02	64.96	65.26	50.20	57.33	59.28	60.99	65.48
	PromDA	**59.37**	**65.65**	69.34	70.06	70.56	61.31	64.00	66.52	67.84	69.18
	Our method	58.90	65.56	**70.86**	**71.51**	**72.75**	**64.14**	**65.38**	**68.29**	**70.09**	**72.31**

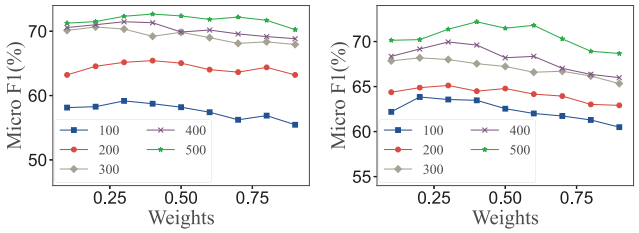
Table 1: Performance of different methods on the OntoNotes and Wikiann datasets under the Shot-{100, 200, 300, 400, 500} settings. The best results are in bold while the second-best ones are underlined.

Model	OntoNotes						Wikiann					
	100		300		500		100		300		500	
	F1(%)	↓%	F1(%)	↓%	F1(%)	↓%	F1(%)	↓%	F1(%)	↓%	F1(%)	↓%
SAINT	58.9	-	70.86	-	72.75	-	64.14	-	68.29	-	72.31	-
w/o syntactic injection	**55.85**	3.05	**66.87**	3.99	**68.91**	3.84	**60.87**	3.27	**64.41**	3.88	68.05	4.26
w/o syntactic loss	57.12	1.78	68.51	2.35	69.78	2.97	61.13	3.01	64.72	3.57	**67.97**	4.34
w/o language model loss	57.57	1.33	69.34	1.52	71.01	1.74	62.85	1.29	66.84	1.45	70.50	1.81

Table 2: Results of the main components ablation experiment, where ↓ represents the model's performance decline. Results with the most significant reduction are marked in bold.



(a) Performance under different syntactic feature injection weights.



(b) Performance under different multi-task learning weights.

Figure 3: Effect of different hyper parameters.

fluency of sentences when $\lambda_{\ell m}$ is large, otherwise, SAINT trends to keep the dependency consistency of sentences. Consequently, SAINT performs better when the relative importance of two tasks is more evenly distributed. Moreover, the trends of the performance with weight $\lambda_{\ell m}$ are roughly the same on both datasets.

3.6 Case Study

In this section, two case studies, i.e., comparing the data quality of each baseline and visualizing the dependency consistency of SAINT, are conducted to help further understand our method.

Comparison of Data Generation

Figure 4 lists the data generated by each method that contains the entity *Bill Gates*. It can be seen that some tokens in the sentences generated by DAGA and MulDA may be *unk*, primarily because these models build their vocabulary using training data, ignoring the special tokens. Furthermore, the sentences generated by DAGA and MulDA are not fluent because they only use a small amount of data to train the language model and do not consider the sentence's grammatical information. PromDA is capable of producing coherent sentences, but the generated tokens are uncontrollable. As a result, *Bill Gates Foundation* in the generated result deviates from the semantics of the raw entity *Bill Gates*. In contrast, on the one hand, SAINT takes syntactic dependency features into account during data generation to ensure the fluency of data generation. On the other hand, SAINT's insertion transformer can help assemble the preset entity list, generating domain-specific sentences. For the sentences generated by SAINT, the entities *Bill Gates*, *Apple*, and *Toyota Motor* are correctly preserved, and the semantics and domain are related to the original sentence.

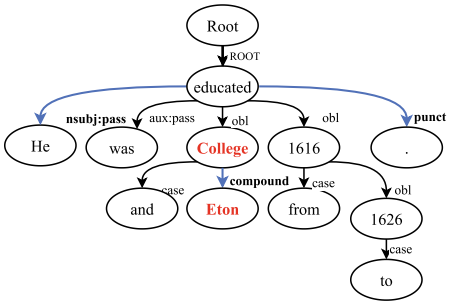
Analysis of Dependency Consistency

Figure 5 presents two dependency trees corresponding to the raw and generated sentences by SAINT. Obviously, the generated sentences maintain a similar syntactic structure to the raw ones. The first observation is that the dependency path between the token *Eton* and *College* are the completely same. Moreover, the dependency paths (-nsubj, obl, compound) and (-nsubj, advmod, conj, compound) between the subject and

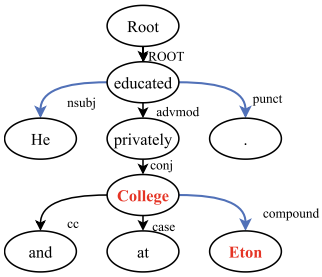
Model	Synthesized Sentence
Raw	[Bill Gates]*person* did not build [Apple]*organization* or [Toyota Motor]*organization*, Japan.
LwTR	L. One not build Apple he Toyota MOT, China.
SR	Bill Gates did non build Apple or Toyota Camry, USA.
MR	Bill Gates did not build Microsoft or Royal Dutch Shell, Japan.
SiS	not build Bill Gates Apple or, Japan Toyota Motor.
DAGA	[Richard unk]*person* was joined in January by [Bill Gates]*person*.
MulDA	It is described by unk of [Bill Gates]*person*.
PromDA	[Bill Gates Foundation]*organization* is established in the United States.
SAINT	[Bill Gates]*person* owns shares of [Apple]*organization* and [Toyota Motor]*organization*.

Figure 4: Generated sentences of LwTR, SR, MR, Sis, DAGA, MulDA, PromDA and SAINT. For editing methods, the tokens in orange are edited. For generative methods, the tokens in blue are the generated entities, and the corresponding types are indicated in italics.

the entity *Eton* are very similar. In conclusion, our method can maintain the raw syntactic dependency, thus encouraging label consistency when generating samples.



(a) Dependency tree of the raw sentence *He was educated at Eton College from 1616 to 1626.*



(b) Dependency tree of the generated sentence *He educated privately and at Eton College*

Figure 5: Dependency trees of the raw and generated sentences. The node marked with red color denotes the organization entity, while the same dependency relations are marked with the same color.

4 Related Work

In this section, we first introduce related research on NER data augmentation, and then focus on controlled text generation which is employed in our work.

4.1 NER Data Augmentation

NER data augmentation can be summarized into two frameworks: editing and generative methods. For the editing methods, Dai et al. and Sabty et al. utilized synonym replacement techniques [Dai and Adel, 2020; Sabty *et al.*, 2021]. Although these methods are simple and easy to understand, the context of augmented samples could be too repetitive. As an alternative, the generative approach uses PLMs to generate samples. Concretely, Chen *et al.* designed a neural network to learn the pattern of text features from raw samples and output synthetic data by transferring such features. Zhou *et al.* introduced an auxiliary classifier into the major model to create data with reversed labels. Liu *et al.* applied a back-translation strategy to address cross-lingual NER. Wang et al. and Liu et al. instructed the PLMs to create samples via prompt learning [Wang *et al.*, 2022; Liu *et al.*, 2022b].

Our method boosts the performance of the NER model with a combination of the insertion-based generation paradigm and the injection of syntactic features to generate more fluent and semantic sentences with better label consistency.

4.2 Controlled Text Generation

Controlled text generation can be divided into autoregressive and non-autoregressive models. For autoregressive generation models, Ma *et al.* developed a simple attention module for GPT-2 [Radford *et al.*, 2019] to strengthen the weight of specific words. Iso produced sentences by an auto-mined template with lexical occurrence limitation. Zhang et al. and Carlsson et al. exploited the novel regulated prompt framework respectively [Zhang and Song, 2022; Carlsson *et al.*, 2022], guiding a causal language model to generate samples for maximum effectiveness. The non-autoregressive model offered some fresh insights into the task. Zhang *et al.* improved the structure of transformer to recursively conduct the insertion process in a parallel manner, thus accelerating the inference process considerably.

The proposed SAINT model explicitly incorporates syntactic features into PLMs and employs a hard constraint generation paradigm to generate text.

5 Conclusion

In this paper, we analyze the main challenges of the low-resource NER data augmentation task and present SAINT, a novel controlled text generation model for NER data augmentation. The proposed method leverages a hard-constrained manner to generate new sentences with lexical and syntactic constraints, which ensures the generated data is consistent with source data both in semantic and syntactic feature patterns. According to the results of experimental results, SAINT can generate more syntactic sentences with fewer noises and improve the performance of the NER model in cases of insufficient annotated data, which exceeds that of the SOTA baseline.

References

[Banerjee *et al.*, 2019] Partha Sarathy Banerjee, Baisakhi Chakraborty, Deepak Tripathi, Hardik Gupta, and Sourabh S Kumar. A information retrieval based on question and answering and ner for unstructured information without using sql. *Wireless Personal Communications*, 2019.

[Cai *et al.*, 2020] Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight. In *Proc. of ACL*, 2020.

[Campos *et al.*, 2020] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Celia Nunes, and Adam Jatowt. Yake! keyword extraction from single documents using multiple local features. *Inf. Sci.*, 2020.

[Carlsson *et al.*, 2022] Fredrik Carlsson, Joey Öhman, Fangyu Liu, Severine Verlinden, Joakim Nivre, and Magnus Sahlgren. Fine-grained controllable text generation using non-residual prompting. In *Proc. of ACL*, 2022.

[Chen *et al.*, 2021] Shuguang Chen, Gustavo Aguilar, Leonardo Neves, and Thamar Solorio. Data augmentation for cross-domain named entity recognition. In *Proc. of EMNLP*, 2021.

[Chen *et al.*, 2022] Xiang Chen, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, Huajun Chen, and Ningyu Zhang. LightNER: A lightweight tuning paradigm for low-resource NER via pluggable prompting. In *Proc. of COLING*, 2022.

[Conneau *et al.*, 2020] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proc. of ACL*, 2020.

[Dai and Adel, 2020] Xiang Dai and Heike Adel. An analysis of simple data augmentation for named entity recognition. In *Proc. of COLING*, 2020.

[de Lichy *et al.*, 2021] Cyprien de Lichy, Hadrien Glaude, and William Campbell. Meta-learning for few-shot named entity recognition. In *Proceedings of the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, 2021.

[Ding *et al.*, 2020] Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proc. of EMNLP*, 2020.

[Gries, 1982] David Gries. A note on a standard strategy for developing loop invariants and loops. *Science of Computer Programming*, 1982.

[Hovy *et al.*, 2006] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: The 90% solution. In *Proc. of AACL*, 2006.

[Iso, 2022] Hayate Iso. Autotemplate: A simple recipe for lexically constrained text generation. *arXiv preprint arXiv:2211.08387*, 2022.

[KERAGHEL *et al.*, 2020] Abdenacer KERAGHEL, Khalid BENABDESLEM, and Bruno CANITIA. Data augmentation process to improve deep learning-based ner task in the automotive industry field. In *Proc. of IJCNN*, 2020.

[Kumar and Byrne, 2004] Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proc. of NAACL*, 2004.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proc. of NAACL*, 2016.

[Liu *et al.*, 2019] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. In *Proc. of CVPR*, 2019.

[Liu *et al.*, 2021] Linlin Liu, Bosheng Ding, Lidong Bing, Shafiq Joty, Luo Si, and Chunyan Miao. MulDA: A multilingual data augmentation framework for low-resource cross-lingual NER. In *Proc. of ACL*, 2021.

[Liu *et al.*, 2022a] Andy T. Liu, Wei Xiao, Henghui Zhu, Dejiao Zhang, Shang-Wen Li, and Andrew Arnold. Qaner: Prompting question answering models for few-shot named entity recognition. *arXiv preprint arXiv:2203.01543*, 2022.

[Liu *et al.*, 2022b] Jian Liu, Yufeng Chen, and Jinan Xu. Low-resource ner by data augmentation with prompting. In *Proc. of IJCAI*, 2022.

[Ma *et al.*, 2022a] Chang Ma, Song Zhang, Gehui Shen, and Zhihong Deng. Switch-gpt: An effective method for constrained text generation under few-shot settings (student abstract). In *Proc. of AAAI*, 2022.

[Ma *et al.*, 2022b] Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Linyang Li, Qi Zhang, and Xuanjing Huang. Template-free prompt tuning for few-shot NER. In *Proc. of NAACL*, 2022.

[Ma *et al.*, 2022c] Tingting Ma, Huiqiang Jiang, Qianhui Wu, Tiejun Zhao, and Chin-Yew Lin. Decomposed meta-learning for few-shot named entity recognition. In *Proc. of ACL Findings*, 2022.

[Min *et al.*, 2020] Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. Syntactic data augmentation increases robustness to inference heuristics. In *Proc. of ACL*, 2020.

[Nallapati *et al.*, 2016] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proc. of CoNLL*, 2016.

[Pan *et al.*, 2017] Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. Cross-lingual name tagging and linking for 282 languages. In *Proc. of ACL*, 2017.

[Peters *et al.*, 2017] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*, 2017.

[Phan and Nguyen, 2022] Uyen Phan and Nhung Nguyen. Simple semantic-based data augmentation for named entity recognition in biomedical texts. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, 2022.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[Sabty *et al.*, 2021] Caroline Sabty, Islam Omar, Fady Wasfalla, Mohamed Islam, and Slim Abdennadher. Data augmentation techniques on arabic data for named entity recognition. *Procedia Computer Science*, 2021.

[Souza *et al.*, 2020] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*, 2020.

[Sun and He, 2020] Xiao Sun and Jiajin He. A novel approach to generate a large scale of supervised data for short text sentiment analysis. *Multimedia Tools and Applications*, 2020.

[Wang *et al.*, 2022] Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. PromDA: Prompt-based data augmentation for low-resource NLU tasks. In *Proc. of ACL*, 2022.

[Wei and Zou, 2019] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proc. of EMNLP*, 2019.

[Wu *et al.*, 2020] Qianhui Wu, Zijia Lin, Guoxin Wang, Hui Chen, Börje F. Karlsson, Biqing Huang, and Chin-Yew Lin. Enhanced meta-learning for cross-lingual named entity recognition with minimal resources. In *Proc. of AAAI*, 2020.

[Yoo *et al.*, 2019] Kang Min Yoo, Youhyun Shin, and Sanggoo Lee. Data augmentation for spoken language understanding via joint variational generation. In *Proc. of AAAI*, 2019.

[Yu *et al.*, 2018] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *Proc. of ICLR*, 2018.

[Zhang and Song, 2022] Hanqing Zhang and Dawei Song. DisCup: Discriminator cooperative unlikelihood prompt-tuning for controllable text generation. In *Proc. of EMNLP*, 2022.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proc. of NeurIPS*, 2015.

[Zhang *et al.*, 2020] Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. POINTER: Constrained progressive text generation via insertion-based generative pre-training. In *Proc. of EMNLP*, 2020.

[Zhou *et al.*, 2022] Jing Zhou, Yanan Zheng, Jie Tang, Li Jian, and Zhilin Yang. FlipDA: Effective and robust data augmentation for few-shot learning. In *Proc. of ACL*, 2022.