DAGA: Data Augmentation with a Generation Approach for Low-resource Tagging Tasks

Bosheng Ding[*12] **Linlin Liu**[*12] **Lidong Bing**[2] **Canasai Kruengkrai**[†3]
Thien Hai Nguyen[2] **Shafiq Joty**[1] **Luo Si**[2] **Chunyan Miao**[1]
[1]Nanyang Technological University, Singapore
[2]DAMO Academy, Alibaba Group [3]National Institute of Informatics, Japan
{bosheng.ding, linlin.liu, l.bing, thienhai.nguyen, luo.si}@alibaba-inc.com
canasai@nii.ac.jp {srjoty, ascymiao}@ntu.edu.sg

Abstract

Data augmentation techniques have been widely used to improve machine learning performance as they enhance the generalization capability of models. In this work, to generate high quality synthetic data for low-resource tagging tasks, we propose a novel augmentation method with language models trained on the linearized labeled sentences. Our method is applicable to both supervised and semi-supervised settings. For the supervised settings, we conduct extensive experiments on named entity recognition (NER), part of speech (POS) tagging and end-to-end target based sentiment analysis (E2E-TBSA) tasks. For the semi-supervised settings, we evaluate our method on the NER task under the conditions of given unlabeled data only and unlabeled data plus a knowledge base. The results show that our method can consistently outperform the baselines, particularly when the given gold training data are less.[1]

1 Introduction

A large amount of training data is often essential for neural model performance, especially, for large networks. Having more training data can help reduce overfitting and improve model robustness. However, preparing a large amount of annotated data is usually costly, labor intensive and time-consuming. Data augmentation (Simard et al., 1998) is a useful technique for synthetic data generation, which is widely used in computer vision (Fawzi et al., 2016; D'Innocente et al., 2017; Wang et al., 2019) and speech (Schlüter and Grill, 2015; Ko et al., 2017).

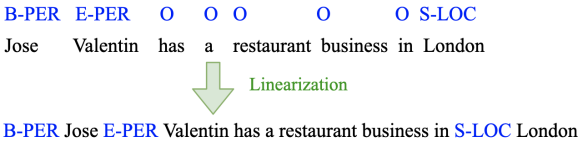

Figure 1: An example of labeled sentence linearization. All words and their tags are paired up by inserting tags before (or after) the words (*O* tags removed).

However, due to the complexity of language, it is more challenging to apply data augmentation techniques to natural language processing (NLP). Unlike computer vision and speech, where handcrafted rules (such as rotation, cropping, masking, etc.) can be easily applied to transform original data, it is difficult to generalize such rules for languages. Although simple distortion usually does not change the semantics of visual information, deleting or replacing a single word could completely change the meaning of the sentence.

One successful method for data augmentation in NLP is *back translation* (Sennrich et al., 2016; Fadaee et al., 2017; Dong et al., 2017; Yu et al., 2018), where a translation model is used to translate monolingual sentences from target language to source language to generate synthetic parallel sentences. Other successful methods include: systematically reordering the dependents of some nodes in gold data to generate synthetic data for dependency parsing (Wang and Eisner, 2016), leveraging knowledge base for question generation (Serban et al., 2016) and using simulation-based approach to generate a set of prerequisite toy tasks for QA (Weston et al., 2015). Besides, synonym replace-

[*]Equal contribution. Bosheng Ding and Linlin Liu are under the Joint PhD Program between Alibaba and Nanyang Technological University.

[†]Work done while at DAMO Academy, Alibaba Group

[1]Our code is available at https://ntunlpsg.github.io/project/daga/

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6045–6057,
November 16–20, 2020. ©2020 Association for Computational Linguistics

ment, random deletion/swap/insertion, generation with VAE or pre-trained language models are also used in some NLP tasks (Kobayashi, 2018; Wei and Zou, 2019; Anaby-Tavor et al., 2020; Raille et al., 2020; Kumar et al., 2020), but mainly for translation and classification tasks.

Compared with the above-mentioned downstream tasks like translation and classification, sequence tagging is more fragile when it is confronted with data augmentation noises due to the finer granularity of the (token-level) task. Annotating unlabeled data with a weak tagger, leveraging aligned bilingual corpora to induce annotation and synonym replacement are three attempted data augmentation methods for sequence tagging (Shang et al., 2018; Yarowsky et al., 2001; Mathew et al., 2019). Weakly labeled data will inevitably introduce more noise. Note that annotating unlabeled data with a weak tagger requires in-domain data and in-domain knowledge, otherwise it may suffer from domain-shift problem (Bari et al., 2020). Leveraging aligned bilingual corpora requires additional resources, which may not be available for low resource languages. Synonym replacement often relies on additional knowledge, e.g., WordNet (Miller, 1995), which is a manually designed dictionary that may have low coverage (or not available) for low-resource languages.

In this work, we investigate data augmentation with a generation approach for sequence tagging tasks. We first linearize the labeled sentences as shown with an example in Figure 1. Then a language model (LM) is trained on the linearized data and used to generate synthetic labeled data. Unlike employing weak taggers to label unseen data, our method unifies the processes of sentence generation and labeling using a LM. Concretely, a word and its tag in a pair (e.g., "B-PER Jose") are trained to be generated together, in which the tag-word pairs with high probability will be chosen by the LM during generation (§3.2). Our method does not require additional resources like WordNet. Nevertheless, if unlabeled data or knowledge bases are available, our method is also flexible to utilize these resources with a simple but effective conditional generation technique (§3.4).

Although some recent work (Anaby-Tavor et al., 2020; Raille et al., 2020; Kumar et al., 2020) also leverages LM for data augmentation, their methods are conditioned on sentence-level tags to generate or modify training data, hence applicable to classification tasks exclusively. To the best of our knowledge, we are the first to utilize generative language models to generate fine-grained synthetic data from scratch for sequence tagging tasks, which introduces a new paradigm for data augmentation in NLP. Furthermore, our method does not rely on large pre-trained models and in fact, it employs a simple one-layer recurrent language model (Sundermeyer et al., 2012), which is more convenient to train. Our method demonstrates encouraging performance when trained on just a few thousand sentences (§4).

To verify the effectiveness of our method, we conduct extensive experiments on different sequence tagging tasks, including named entity recognition (NER), part-of-speech (POS) and end-to-end target based sentiment analysis (E2E-TBSA). Our method consistently outperforms the baseline methods in both supervised and semi-supervised settings. Different from the baseline methods, our method generates novel synthetic data from scratch, and thus introduces more diversity to reduce overfitting. For the semi-supervised settings, our method demonstrates strong ability to exploit useful information from unlabeled data and knowledge base.

2 Background

Named Entity Recognition (NER) Named entities refer to phrases that are names of persons, organizations and locations, etc. in text. For example, "*[ORG U.N.] official [PER Ekeus] heads for [LOC Baghdad]*". Named entity recognition is an important task of information extraction and it aims to locate and classify named entities in text into the predefined types (Mikheev et al., 1999; Sang and De Meulder, 2003; Li et al., 2020). It is a challenging task for two reasons (Lample et al., 2016): 1) in most languages and domains, the amount of manually labeled training data for NER is limited; 2) it is difficult to generalize from this small sample of training data due to the constraints on the kinds of words that can be names.

Part-of-Speech (POS) Tagging Part-of-speech tagging consists of assigning a tag that represents a grammatical class to each word in a given sentence. It is a critical component of most NLP systems and is fundamental to facilitate downstream tasks such as syntactic parsing (Schütze, 1993) and opinion analysis (Liu et al., 2015). The current state-of-the-art POS taggers can achieve over 97.80% accuracy on PTB-WSJ (Akbik et al., 2018; Bohnet

et al., 2018) and yield over 96.50% average test accuracy across 21 high-resource languages in UD 1.2 (Heinzerling and Strube, 2019). However, one of the problems with current POS taggers is that their accuracy can decrease significantly on low-resource languages and rare words (Plank et al., 2016; Yasunaga et al., 2018).

Target Based Sentiment Analysis The target based sentiment analysis is a fundamental task of sentiment analysis and it aims to detect the opinion targets in sentences and predict the sentiment polarities over the targets (Liu et al., 2015; Chen et al., 2017; Li et al., 2018, 2019a). For example, *"**USB3 Peripherals** are noticeably less expensive than the **ThunderBolt ones**"*. In this sentence, two opinion targets were mentioned, namely "***USB3 Peripherals***" and "***ThunderBolt ones***" and the user expresses a positive sentiment over the first, and a negative sentiment over the second. Li et al. (2019a,b) propose an end-to-end solution (E2E-TBSA) of TBSA, which converts TBSA to a tagging task, and aims to solve the two subtasks (i.e. target detection and sentiment classification) in a unified manning by predicting unified tags. For example, the tag "B-POS" indicates the beginning of a target with positive sentiment. So after annotation, the above example becomes *"[B-POS USB3] [E-POS Peripherals] are noticeably less expensive than the [B-NEG ThunderBolt] [E-NEG ones]"*.

3 Proposed Method

We propose a novel data augmentation method for sequence tagging tasks. We first linearize labeled sentences, and then train a language model to learn the distribution of words and tags from the linearized sequences for generating synthetic training data. A conditional generation technique is also proposed to exploit unlabeled data and knowledge bases when they are available.

3.1 Labeled Sentence Linearization

We first perform sentence linearization to convert labeled sentences into linear sequences, so that language models can be used to learn the distribution of words and tags in gold data. As shown in Figure 1, tags are inserted before the corresponding words during linearization and thus treated as modifiers of these words. For the tasks with frequent *O* tags, e.g., NER and E2E-TBSA (Li et al., 2019a), we remove such tags from the linearized sequences.

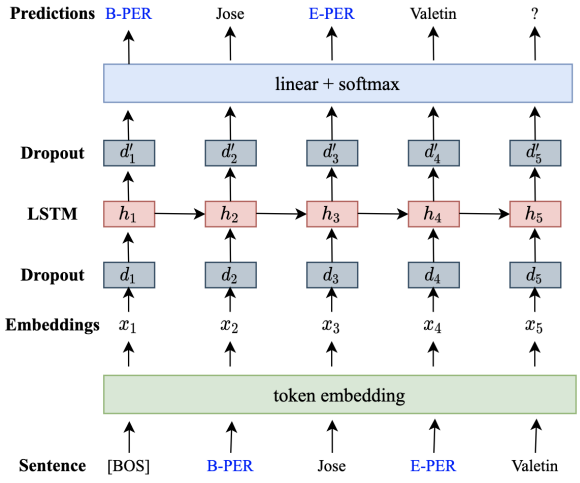

Figure 2: Language model architecture with LSTM.

Similarly, we can also insert tags after the corresponding words.

After sentence linearization, we add special tokens *[BOS]* and *[EOS]* to the beginning and the end of each sentence, respectively. These special tokens are used to facilitate model training and data generation by marking the sentence boundaries.

3.2 Language Modeling and Data Generation

After linearizing labeled sentences, language models can be used to learn the distribution of words and tags. More specifically, we use a one-layer LSTM recurrent neural network language model (RNNLM) in our method, which is similar to the model proposed by Sundermeyer et al. (2012). The architecture of our RNNLM is shown in Figure 2.

Language Modeling We train RNNLM by maximizing the probability for next token prediction. Given a sentence, we first feed the sequence of tokens $(w_1, w_2, \ldots, w_N)$ into the embedding layer to lookup the corresponding embeddings $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, where N is the sequence length. A dropout layer is applied to each token embedding $\boldsymbol{x}_t$ to generate $\boldsymbol{d}_t = \mathrm{dropout}(\boldsymbol{x}_t)$. Then we feed $(\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_N)$ into LSTM to produce hidden state $\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{d}_t, \boldsymbol{h}_{t-1})$ at each position t. Another dropout layer is applied to hidden states to compute $\boldsymbol{d}'_t = \mathrm{dropout}(\boldsymbol{h}_t)$.

Finally, a linear+softmax layer is used to predict the next token in the sequence. Assuming the index of token w_t in the vocabulary is i^*, we have the

[BOS] [labeled] B-PER Jose E-PER Valentin has a restaurant business in S-LOC London [EOS]

[BOS] [unlabeled] I have booked a flight to New York [EOS]

[BOS] [KB] We ate crepes in S-LOC Shibuya, saw cherry blossom bloom at Asakusa [EOS]

Figure 3: An example of conditional generation. The first sequence is from gold NER data. The second is from unlabeled data, so no labels. The third is labeled by knowledge base matching, where *Asakusa* cannot be labeled due to incomplete knowledge coverage.

training objective in Eq. 3:

$$\boldsymbol{s}_{t-1} = \boldsymbol{M}^{\top} \boldsymbol{d}'_{t-1} \quad (1)$$

$$p_\theta(w_t|w_{<t}) = \frac{\exp(\boldsymbol{s}_{t-1,i^*})}{\sum_{i=1}^{V} \exp(\boldsymbol{s}_{t-1,i})} \quad (2)$$

$$p(w_1, w_2, \ldots, w_N) = \prod_{t=1}^{N} p_\theta(w_t|w_{<t}) \quad (3)$$

where V is the size of vocabulary, $\boldsymbol{M} \in \mathbb{R}^{r \times V}$ is a learnable weight matrix with r being the dimension of LSTM hidden states, and $\boldsymbol{s}_{t-1,i}$ is the i-th element of $\boldsymbol{s}_{t-1}$.

Generation After training the RNNLM, we can use it to generate synthetic training data for tagging tasks. During generation, only the *[BOS]* token is fed into RNNLM, and the following tokens are sampled based on the probabilities computed by Eq. 2. Given *[BOS]*, the sentence is generated autoregressively one at a time, where the token generated in the previous step is taken as input to generate the next one.

As shown in Eq. 2, RNNLM is more likely to pick the tokens with high probabilities during sampling in the generation process. Because of the randomness added by sampling, RNNLM can choose similar alternatives given the same context. Assume we insert tags before the corresponding words (during sentence linearization) to train the RNNLM, when predicting the next token given *"I have booked a flight to"*, the probability of *"S-LOC"* is much higher than the other choices, since the RNNLM has seen many similar examples in training data, such as *"a train to S-LOC"*, *"a trip to S-LOC"* and so on. Then we predict the following word given *"I have booked a flight to S-LOC"*. In the training data, all *"S-LOC"* are followed by location words, so *"London"*, *"Paris"*, *"Tokyo"*, etc., are all possible choices, and their probabilities are very close. Due to the added randomness, the model can choose any one of them. Tokens are predicted in a similar way when we insert tags after the corresponding words, except that words are predicted before the tags.

3.3 Post-Processing

The generated sequences are in the linearized format, so they need to be converted to the same format as the gold data. We also introduce several straightforward rules to clean the generated data: 1) Delete sentences with no tags; 2) Delete sentences where all words are *[unk]*[2]; 3) Delete sentences with incorrect tag prefix orders (e.g., having *E-LOC* before *B-LOC* in NER data); 4) Delete sentences that contain same sequences of words but different tags.

3.4 Conditional Generation

We propose a conditional generation method to allow the language model to utilize unlabeled data or knowledge bases when they are available in some low-resource scenarios. For example, it could be expensive to annotate a large amount of e-commerce product titles for NER, but much easier to obtain a knowledge base (i.e., dictionary) of product attributes and unlabeled data. We prepend one of these **condition tags** $\{$*[labeled]*, *[unlabeled]*, *[KB]*$\}$ at the beginning of each sequence to mark their origin, where *KB* means the sequence is labeled by matching a knowledge base against the unlabeled data. See Figure 3 for an example. This allows the language model to learn the shared information among these sequences while being aware of the different origins. When generating synthetic data, each word is conditioned on the given condition tag *[labeled]*, denoted as c (conditioning class). After we feed it into the language model, all the LSTM hidden states $\boldsymbol{h}$ in the following generation steps contain information of c. In addition, $\boldsymbol{h}$ also encodes information of all of the other previous tokens in

[2]To reduce the size of vocabulary when training language model, the words that only appear once in training data are replaced with the unknown token *[unk]*.

the sequence. When predicting the next token conditioned on $\boldsymbol{h}_{t-1}$, the probability $p_\theta(w_t|w_{<t})$ in Eq. 2 becomes $p_\theta(w_t|w_{<t}, c)$.[3] A similar approach is used in CTRL (Keskar et al., 2019) to control style, task-specific behavior, etc., during text generation.

4 Experiments

In this section, we present our experiments in both supervised and semi-supervised settings. In the supervised settings, only gold data are used for augmentation. In the semi-supervised settings, we also leverage unlabeled data and knowledge bases.

4.1 Basic Models

Language Model We use the language model described in Section 3.2 for synthetic data generation. We modified the decoder of the LSTM-LM model in Kruengkrai (2019) to implement this language model. We set LSTM hidden state size to 512 and embedding size to 300. We use dropout rate 0.5 for the two dropout layers. All language model are trained using Stochastic gradient descent (SGD) with initial learning rate 1 and batch size 32. Learning rate will be decayed by 0.5 in the next epoch if the perplexity on dev set does not improve. We set the maximum number of epochs to 30 and stop training early if the perplexity on dev set does not improve in 3 consecutive epochs. During synthetic data generation, we use the average length of gold sentences in the training set as our maximum sentence length.

Sequence Tagging Model We implement a BiLSTM-CRF model (Lample et al., 2016) with the Flair framework (Akbik et al., 2019) to evaluate our data augmentation method on NER and POS tasks.[4] We use a single-layer BiLSTM with hidden state size 512. Dropout layers are applied before and after the BiLSTM layer with dropout rate 0.5. All sequence tagging models are trained using Adam (Kingma and Ba, 2014) with initial learning rate 1e-3 and batch size 32. Learning rate is decayed by 0.5 if the performance on dev set does not improve in 3 consecutive epochs. We stop training when the learning rate drops below 1e-5 or number of epochs reaches 100. We use the pre-trained 300-dimensional fastText word embeddings (Bojanowski et al., 2017) for all languages.

We employ relatively simple basic models because: 1) They help to avoid the possible overfitting problems due to the small data size under the low resource setting; 2) They allow more faithful understanding on the effects of the proposed data augmentation method.

4.2 Supervised Experiments

To verify the effectiveness of our data augmentation method in the supervised settings, we evaluate it on three different tagging tasks, including NER, POS and E2E-TBSA. Most of the prior works rely on additional information, so we use random deletion (**rd**) (Wei and Zou, 2019) as our baseline, where 5% of the words[5] and the corresponding tags in training data are randomly deleted. See Table 1 for the notations of the methods used in our supervised experiments.

Method	Description
gold	Only use the gold data.
gen	Our method. Generate synthetic data with the language models, and oversample gold data.
rd	Baseline method. Generate synthetic data by random deletion, and oversample gold data with the same ratio as **gen**.
rd*	Baseline method. Similar to **rd**, except that gold and synthetic data are equally sampled.

Table 1: Data sources for the supervised setting.

4.2.1 Named Entity Recognition

Dataset We evaluate our proposed methods on the CoNLL2002/2003 NER data (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), with four languages: English, German, Dutch and Spanish. Besides, we evaluate our methods on Thai and Vietnamese NER data, which are product titles obtained from major e-commerce websites in Southeast Asian countries and annotated with 11 product attribute NER tags, including *PRODUCT*, *BRAND*, *CONSUMER_GROUP*, *MATERIAL*, *PATTERN*, *COLOR*, *FABRIC*, *OCCASION*, *ORIGIN*, *SEASON* and *STYLE*. See Appendix for the statistics of the Thai and Vietnamese NER data used in our experiments.

Experimental Settings In addition to evaluating our method on the full training data, we also ran-

[3] Condition tag c is also in $w_{<t}$, since it is a special token added to the beginning of each sentence. We write it explicitly to emphasize the conditional effect.

[4] The baseline model provided in the original paper is used for evaluating end to end target based sentiment analysis task.

[5] For NER and E2E-TBSA, the whole entity is deleted if a selected word appears within an entity span.

domly sample 1k, 2k, 4k, 6k and 8k sentences for each language to verify its robustness in low-resource settings. We use the original development and test data. The language models are trained on the above sampled sentences, and then used to generate synthetic training data following the steps described in Section 3.

For each new batch of 1k sentences generated by the trained language model, we measure the percentage of new 1-gram tokens that appears in previous batches. Once the percentage exceeds 99%, we will stop data generation. Then we post-process (Section 3.3) the generated data, and add them to gold training data for tagging model training. For **rd** and **gen**, we oversample gold data by repeating them 4 times (shuffled) in the training set. For random deletion, we also report the result when gold and synthetic data are equally sampled, denoted by **rd***. Additional details on hyperparameter selection for the oversampling ratios can be found in Appendix A.3. Following Lample et al. (2016), the IOBES tagging scheme is used when training the language models and sequence tagging models described above.

Results and Analysis We report the average results of 3 runs in Table 2. Our method shows consistent performance improvement for all languages. Especially for the smaller sampled sets, our method demonstrates more significant performance improvement. In particular, the proposed method achieved average 1.93 and 1.38 point improvement compared with the baseline methods in the 1k and 2k settings, respectively.

Tag-Word vs. Word-Tag As discussed in Section 3.1, there are two ways to perform sentence linearization: 1) insert tags before the corresponding words (Tag-Word); 2) insert tags after the corresponding words (Word-Tag). Keeping all of the other settings same, we find Tag-Word outperforms Word-Tag in NER tasks (as shown in Appendix A.2). One possible reason is that, Tag-Word is more consistent with the Modifier-Noun pattern, which appears more often in the training data during language modeling. Therefore, we use Tag-Word for all the NER experiments.

[6]German has large out-of-vocabulary rate when using static fastText embedding, which leads to lower F1 score compared with other languages.

Lang.	Method	1k	2k	4k	6k	8k	all
en	gold	58.06	67.85	74.55	77.16	80.30	83.04
	+rd*	59.42	67.23	74.51	77.39	80.31	83.39
	+rd	58.97	67.81	74.77	77.35	80.59	83.25
	+gen	**61.15**	**70.61**	**76.82**	**79.18**	**81.02**	**83.74**
de[6]	gold	29.71	**41.07**	49.55	53.30	56.17	61.10
	+rd*	29.89	40.29	49.27	52.33	55.70	60.69
	+rd	30.83	40.36	49.24	53.54	55.60	60.55
	+gen	**31.83**	40.92	**49.79**	**53.63**	**56.94**	**62.44**
es	gold	58.14	67.42	74.21	77.44	78.90	79.27
	+rd*	58.22	66.98	75.08	77.64	79.11	80.01
	+rd	59.67	68.53	75.21	77.79	79.12	80.26
	+gen	**61.76**	**68.62**	**76.15**	**78.20**	**79.83**	**80.73**
nl	gold	37.04	48.61	57.78	61.08	64.59	70.89
	+rd*	35.10	46.45	56.83	60.49	63.09	69.42
	+rd	**39.39**	48.44	59.38	61.48	64.44	70.36
	+gen	38.87	**50.41**	**59.90**	**63.19**	**65.82**	**72.71**
vi	gold	55.98	62.42	69.01	70.75	72.12	76.14
	+rd*	55.67	63.57	68.47	70.87	72.08	76.43
	+rd	56.24	63.08	68.63	71.15	72.22	76.83
	+gen	**60.01**	**65.43**	**70.36**	**72.55**	**74.11**	**77.39**
th	gold	49.88	55.79	61.75	63.10	64.94	67.71
	+rd*	50.46	56.98	62.12	64.19	66.47	67.81
	+rd	50.52	57.42	61.51	64.59	66.07	67.97
	+gen	**54.02**	**59.36**	**63.94**	**66.21**	**68.05**	**69.86**

Table 2: Named entity recognition micro F1.

4.2.2 Part of Speech Tagging

Dataset We use the POS data from Universal Dependencies treebanks[7] for evaluation on this task. We evaluate on five languages, including English, Spanish, Czech, Romanian and Japanese. Each language has multiple corpora in the Universal Dependencies treebanks, so we merge the corpora to build one dataset for three languages: English, Spanish and Czech. For English, we merge *GUM*, *ParTUT*, *PUD* and *Lines*. For Spanish, we merge *AnCora* and *GSD*. For Czech, we merge *PDT*, *FicTree*, *CLTT* and *CAC*. We have also evaluated our model on Japanese (*GSD*) and Romanian (*RRT*), which are either spoken by a much smaller population or from a different language family.

Settings and Results We follow similar experimental settings as NER task. The same language model and BiLSTM-CRF sequence tagging model are used for synthetic data generation and POS tagging respectively. Different from NER, Word-Tag shows slightly better performance in POS tasks (refer to Appendix A.2). We present the average Word-Tag results of 3 runs in Table 3. Our method demonstrates consistent performance improvement for all languages. Similar to NER, our method demon-

[7]https://universaldependencies.org/

Lang.	Method	1k	2k	4k	6k	8k	Full
en	gold	79.18	82.17	85.83	88.62	90.21	93.00
	+rd*	79.28	82.42	85.82	88.55	90.07	92.89
	+rd	79.38	82.50	86.08	88.80	90.15	92.96
	+gen	**79.76**	**82.90**	**86.31**	**88.99**	**90.56**	**93.29**
es	gold	88.28	90.79	92.82	93.80	94.43	96.40
	+rd*	88.25	90.94	92.84	93.76	94.48	96.41
	+rd	88.17	90.78	92.79	93.67	94.28	**96.45**
	+gen	**88.77**	**91.04**	**93.12**	**93.93**	**94.64**	**96.45**
cz	gold	80.10	84.46	88.88	90.67	92.03	97.52
	+rd*	79.83	84.29	88.64	90.43	91.95	97.57
	+rd	80.11	84.50	88.99	90.66	91.86	97.60
	+gen	**80.65**	**85.17**	**89.58**	**91.22**	**92.49**	**97.63**
ro[8]	gold	86.69	89.57	92.73	93.84	94.54	94.54
	+rd*	86.42	89.58	92.50	93.89	94.64	94.64
	+rd	86.62	89.46	92.55	93.84	94.73	94.73
	+gen	**87.29**	**90.66**	**93.44**	**94.61**	**95.17**	**95.17**
ja[9]	gold	90.19	91.44	93.59	94.41	-	95.08
	+rd*	90.00	91.41	93.66	94.62	-	94.93
	+rd	89.53	91.76	93.62	94.59	-	95.18
	+gen	**91.00**	**92.51**	**94.12**	**95.21**	-	**95.45**

Table 3: POS tagging accuracy.

strates more significant performance improvement for smaller sampled sets on POS tagging. In particular, the proposed method achieved average 0.56, 0.60 and 0.46 point improvement compared with the baseline methods in the 1k, 2k and 4k settings, respectively.

4.2.3 Target Based Sentiment Analysis

Dataset We use the laptop and restaurant review datasets processed by Li et al. (2019a) for evaluation on E2E-TBSA, which was initially obtained from SemEval ABSA challenges (Pontiki et al., 2014, 2015, 2016). We merge these two review datasets, regard 10% randomly held-out training data as the dev set, and randomly sample smaller sets from the remaining training data for low-resource settings. The original test sets are merged as our test set.

Settings and Results We follow similar experimental settings as NER and POS tasks, except that the same sequence tagging model released by Li et al. (2019a) is used for evaluation. Here Tag-Word shows better results (refer to Appendix A.2), plausibly it is because the unified tags (e.g. *B-POS*, and *B-NEG*) are similar to noun modifiers and Tag-Word is more consistent with the Modifier-Noun pattern. We present the average Tag-Word results of 3 runs in Table 4. Our method demonstrates performance improvement for 4k and above. Compared with the NER and POS datasets, the E2E-TBSA dataset has much fewer labels, so the results are less stable.

Method	2k	4k	all(6k)
gold	56.31	60.43	63.18
+rd*	**57.92**	61.75	63.66
+gen	57.07	**62.66**	**65.86**

Table 4: E2E-TBSA micro F1.

4.3 Semi-supervised Experiments

In this section we evaluate the effectiveness of our method in two semi-supervised settings: 1) only unlabeled data are available; 2) both unlabeled data and knowledge base are available. See Table 5 for the notations of the methods used in our semi-supervised experiments.

Method	Description
gold	Supervised method. Only use the gold data.
wt	Baseline method. Annotate unlabeled data with a weak tagger (i.e. a tagging model trained on the gold data).
gen$_{ud}$	Our method. Generate synthetic data with LM, where LM is trained on gold data and unlabeled data.
kb	Baseline method. Annotate unlabeled data with knowledge base.
gen$_{kb}$	Our method. Generate synthetic data with LM, where LM is trained on gold data and knowledge base annotated data.

Table 5: Data sources for the semi-supervised setting.

4.3.1 Only Using Unlabeled Data

Dataset We use CoNLL2003 English NER data (Tjong Kim Sang and De Meulder, 2003) for evaluation. In addition to the gold NER training data, we utilize unlabeled data for semi-supervised training. The Stanford CoreNLP tokenizer (Manning et al., 2014) is used to tokenize Wikipedia sentences.

Experimental Settings Similar to the above experiments, we use 1k, 2k, 4k, 6k and 8k sentences randomly sampled from NER gold data as well as the full dataset to evaluate our method. For fair comparison, we only use the same set of 10k sentences randomly sampled from Wikipedia dump in both of our and baseline methods. Let D_{gold} and $D_{unlabeled}$ be the sampled gold NER data and the Wikipedia data, respectively.

In our method, D_{gold} and $D_{unlabeled}$ are concatenated to train language models, following the steps

[8] UD-RRT full train set has 8k sentences for Romanian.
[9] UD-GSD full train set has 7k sentences for Japanese.

Method	1k	2k	4k	6k	8k	all
gold	58.06	67.85	74.55	77.16	80.30	83.04
+wt	65.12	72.43	77.90	79.41	81.36	84.00
+gen$_{ud}$	**66.19**	**73.00**	**78.08**	**79.75**	**81.98**	**84.33**
+kb	**67.36**	72.86	77.15	79.33	81.91	83.69
+gen$_{kb}$	66.67	**73.54**	**78.32**	**79.98**	**81.93**	**84.03**

Table 6: Semi-supervised NER F1.

described in Section 3.4. Then we use the language models to generate synthetic data, from which 20k randomly sampled sentences are combined with D_{gold} to train NER models. We use **gen$_{ud}$** to denote this data generation method. The method that employs weak taggers to annotate $D_{unlabeled}$ is used as our baseline, denoted by **wt**. The weak taggers in this experiment are the NER models trained on D_{gold}. We use the same NER model (BiLSTM-CRF) and hyperparameters to evaluate our and baseline methods. When training the language model, we equally sample sentences from D_{gold} and $D_{unlabeled}$. When training the NER models, we oversample the gold data by repeating D_{gold} 4 times to create a shuffled training file.

Results and Analysis We report F1 of **wt** and **gen$_{ud}$** (average of 3 runs) in Table 6. Our method outperforms the baseline method **wt** on all settings. Moreover, it would be a promising direction to further explore our model's capability by using a larger amount of unlabeled data. It is not very convenient to utilize a large amount of unlabeled data in the baseline method **wt**, since this will directly increase the amount of augmented data. As a result, some of augmented data may not be utilized before sequence tagging models converge. However, our method **gen$_{ud}$** can conveniently utilize a large amount of unlabeled data to train the language models, thereby improving data augmentation quality directly. When the amount of unlabeled data is much larger, we can pretrain language models with the unlabeled data, and then finetune them with labeled data.

4.3.2 Using Unlabeled Data and Knowledge Base

Dataset In addition to the gold training data and unlabeled sentences used in Section 4.3.1, we also try to leverage knowledge base for further performance improvement in this experiment. We build the knowledge base by extracting entities (case sensitive and appearing at least twice) and the corresponding tags from the full gold training data. Besides, we add more *LOC* entities to this knowledge base by including the cities and countries extracted from geonames[10].

Experimental Settings We randomly sample the gold NER data and the Wikipedia data in the same way as Section 4.3.1, where the sampled sentences are denoted by D_{gold} and $D_{unlabeled}$, respectively. Our knowledge base is used to annotate $D_{unlabeled}$ by finding longest forward matches (from left to right) in each sentence. We denote this annotation method by **kb** and the annotated data by D_{kb}.

In our method, D_{gold} and D_{kb} are concatenated to train language models, following the steps described in Section 3.4. Then we use the language models to generate synthetic data, from which 20k randomly sampled sentences are combined with D_{gold} to train the NER models. We use **gen$_{kb}$** to denote this data generation method and compare it with the baseline method **kb**. Similar to the above experiments, we oversample D_{gold} when training the language and NER models.

Results and Analysis We present F1 of **kb** and **gen$_{kb}$** (average of 3 runs) in Table 6. The baseline method **kb** exhibits very strong performance when the size of D_{gold} is small, since we use a large knowledge base of countries and cities for annotation, and location names are less ambiguous compared with the other types of entities. However, our method still outperforms **kb** when the size of D_{gold} is larger than 2k, which shows that our method is more robust to the noises in D_{kb} when a slightly larger amount of gold data are available.

5 A Closer Look at Synthetic Data

In this section, we explore in more details why the synthetic data generated by our method can help improve sequence tagging performance. Through a closer look at the generated data, we have several interesting findings.

More Diversity The generated synthetic data introduces more diversity to help reduce overfitting. As the example shown in Figure 4, the name "*Sandrine*" in the gold training data always pairs up with "*Testud*" in different sentences. However, in the generated data, we can see new names have been generated like "*Sandrine Nixon*", "*Sandrine Okuda*" and "*Sandrine Neuumann*". Meanwhile,

[10]https://datahub.io/core/world-cities

Gold Training Data

1. ... [B-PER Sandrine] [E-PER Testud] ([S-LOC France]) beat ...
2. ... [B-PER Sandrine] [E-PER Testud] ([S-LOC France]) ...
3. ... beat [B-PER Sandrine] [E-PER Testud] ([S-LOC France]) ...
4. ... [B-PER Sandrine] [E-PER Testud] ([S-LOC France]) beat ...

Generated Data

1. ... [B-PER Sandrine] [E-PER Testud] ([S-LOC Sweden]) ...
2. ... [B-PER Sandrine] [E-PER Nixon] fled to ([S-LOC Egypt]) ...
3. ... [B-PER Sandrine] [E-PER Okuda] ([S-LOC Australia]) ...
4. ... [B-PER Sandrine] [E-PER Neuumann] ([S-LOC France]) beat ...

Figure 4: An illustration of diversity of generated data. The name "*Sandrine*" in the gold training data always pairs up with "*Testud*" in sentences.

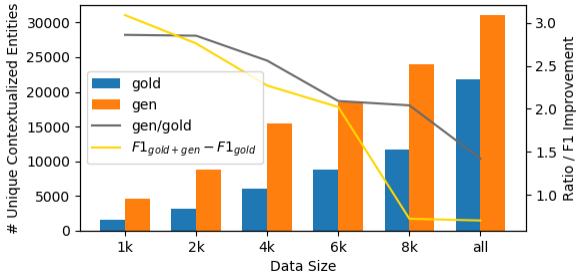

Figure 5: Statistics of unique contextualized entities.

the locations in the sentences have been replaced with new countries like "*Sweden*", "*Egypt*" and "*Australia*". With these synthetic data, the model can focus on learning the pattern of contexts that entities appear in, instead of simply memorizing "*Sandrine Testud*" as a person name and "*France*" as a location.

To quantitatively measure the diversity brought in by our method and its impact, we have done some statistical analysis of the contextualized entities (CEs) in the generated data of the supervised English NER. A CE refers to the combination of an entity and its 1-gram contexts. For example, in the sentence "*The [B-ORG European] [E-ORG Commission] said ...*", the entity is "*European Commission*" and its CE is "*The European Commission said*". As the shown in Figure 5, we calculate the number of unique CEs in the gold training data, the number of new unique CEs in the generated data, and the ratio of the two numbers. We also plot here the F1 improvement of our method (i.e. gold+gen) over only using the gold data, as given in Table 2. We can see that our method generates a large number of new CEs, and such diversity strengthens the robustness of the trained model. When the ratio is higher, the F1 improvement is more significant, which also shows that our method does help ease the low-resource problem by generating rich new entities and contexts. Refer to the Appendix A.5 for statistics of unique entities (without context), which shows the same conclusion.

Efficient Utilization of Unlabeled Data When unlabeled data are available, our method is flexible to utilize them for semi-supervised training. We find many interesting examples in the synthetic data that show our method can effectively use the unlabeled data to extract useful information. In an example generated by our method "... *the [B-ORG Bank] [I-ORG of] [E-ORG Alabama] ...*", the word "*Alabama*" has never appeared in gold NER training data. However, our language model learned that "*Alabama*" (from unlabeled data) is very similar to the other location words that appear in both gold training data and unlabelled data. So when generating the synthetic data, the language model can use this word in a similar context, or even create new entities ("*Bank of Alabama*" in this example has never appeared in gold or unlabeled data).

6 Conclusion

In this paper, we show that language models can be used to generate high quality synthetic data for sequence tagging tasks. The generated data introduce more diversity to reduce overfitting, since they are generated from scratch instead of modifying the gold training data. Our proposed method demonstrates promising performance improvements on various tagging tasks, especially in the low-resource settings. Besides, experiments demonstrate that our method can also effectively utilize unlabeled data and knowledge base for semi-supervised training.

Acknowledgements

This research is partly supported by the Alibaba-NTU Singapore Joint Research Institute, Nanyang Technological University. Linlin Liu would like to thank the support from Interdisciplinary Graduate School, Nanyang Technological University.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Com-*

putational Linguistics (Demonstrations), pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Ateret Anaby-Tavor, 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 *Thirty-Third AAAI Conference on Artificial Intelligence*.

M Saiful Bari, Tasnim Mohiuddin, and Shafiq Joty. 2020. Multimix: A robust data augmentation framework for cross-lingual nlp.

Bernd Bohnet, Ryan McDonald, Gonçalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. 2018. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of EMNLP*, pages 463–472.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, Copenhagen, Denmark. Association for Computational Linguistics.

Antonio D'Innocente, Fabio Maria Carlucci, Mirco Colosi, and Barbara Caputo. 2017. Bridging between computer and robot vision through data augmentation: a case study on object recognition. In *International Conference on Computer Vision Systems*, pages 384–393. Springer.

Marzieh Fadaee, 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 (Volume 2: Short Papers)*, pages 567–573.

Alhussein Fawzi, Horst Samulowitz, Deepak Turaga, and Pascal Frossard. 2016. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692. Ieee.

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

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. 2017. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224. IEEE.

Sosuke Kobayashi. 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.

Canasai Kruengkrai. 2019. Better exploiting latent variables in text modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5527–5532.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Xin Li, Lidong Bing, Piji Li, and Wai Lam. 2019a. A unified model for opinion target extraction and target sentiment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6714–6721.

Xin Li, Lidong Bing, Piji Li, Wai Lam, and Zhimou Yang. 2018. Aspect term extraction with history attention and selective transformation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4194–4200.

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019b. Exploiting BERT for end-to-end aspect-based sentiment analysis. In *Proceedings of the*

5th Workshop on Noisy User-generated Text, W-NUT@EMNLP, pages 34–41.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, Lisbon, Portugal. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Joel Mathew, Shobeir Fakhraei, and José Luis Ambite. 2019. Biomedical named entity recognition via reference-set augmented bootstrapping. *arXiv preprint arXiv:1906.00282*.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 1–8. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, San Diego, California. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.

Guillaume Raille, Sandra Djambazovska, and Claudiu Musat. 2020. Fast cross-domain data augmentation through neural sentence editing. *arXiv preprint arXiv:2003.10254*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Jan Schlüter and Thomas Grill. 2015. Exploring data augmentation for improved singing voice detection with neural networks. In *ISMIR*, pages 121–126.

Hinrich Schütze. 1993. Part-of-speech induction from scratch. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 251–258. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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, Berlin, Germany. Association for Computational Linguistics.

Iulian Vlad Serban, Alberto Garcia-Duran, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837.

Patrice Y. Simard, Yann A. LeCun, John S. Denker, and Bernard Victorri. 1998. *Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation*, pages 239–274. Springer Berlin Heidelberg, Berlin, Heidelberg.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *Transactions of the Association for Computational Linguistics*, 4:491–505.

Xiang Wang, Kai Wang, and Shiguo Lian. 2019. A survey on face data augmentation. *arXiv preprint arXiv:1904.11685*.

Jason Wei 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 6383–6389.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. Technical report, JOHNS HOPKINS UNIV BALTIMORE MD DEPT OF COMPUTER SCIENCE.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 976–986.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*.

A Appendix

A.1 Statistics of Thai and Vietnamese NER Data

We present the number of sentences in Thai and Vietnamese NER data in Table 7.

Lang.	train	dev	test
vi	18,922	500	500
th	11,272	499	490

Table 7: Number of sentences in TH and VI NER data.

A.2 Experiments on Tag-Word vs. Word-Tag

We conduct experiments to compare the performance of Tag-Word and Word-Tag for the tagging tasks. All of the other settings are same as the corresponding experiments presented in the main paper. Results are reported in Table 8 to 10. Tag-Word yields better average performance for NER and E2E-TBSA, while Word-Tag slightly outperforms Tag-Word for POS tagging.

Lang.	Method	1k	2k	4k	6k	8k	full	average
en	Tag-Word	**59.39**	**69.48**	**75.68**	**78.65**	**80.19**	**83.70**	**74.52**
	Word-Tag	58.97	67.32	75.45	78.06	80.43	83.58	73.97

Table 8: CoNLL NER F1: Tag-Word vs. Word-Tag.

Lang.	Method	1k	2k	4k	8k	15k	average
en	Tag-Word	79.06	82.43	85.93	**90.38**	**92.75**	86.11
	Word-Tag	**79.18**	**82.64**	**86.13**	90.33	92.68	**86.19**

Table 9: Universal Dependencies POS accuracy: Tag-Word vs. Word Tag.

Lang.	Method	2k	4k	full(6k)	average
en	Tag-Word	54.22	**61.72**	**62.88**	**59.61**
	Word-Tag	**55.58**	59.42	61.65	58.88

Table 10: E2E-TBSA micro F1: Tag-Word vs. Word-Tag.

A.3 Experiments on Oversampling Ratios

We conduct experiments to compare different oversampling ratios for NER task. Results are reported in Table 11. The notation gold$\times N$ means we oversample gold by repeating it N times in the shuffled static training data.

A.4 Semi-supervised Experiments on Part of Speech Tagging

Dataset We use the English POS data from Universal Dependencies treebanks for evaluation on this task. We merge *GUM*, *ParTUT*, *PUD* and *Lines* corpora to build the English dataset. Similar to the semi-supervised experiments on NER, we also utilize unlabeled Wikipedia sentences for training.

Experimental Settings We use 1k, 2k, 4k, 6k and 8k sentences randomly sampled from English POS gold data as well as the full dataset to evaluate our method. We follow the same experimental setting as the semi-supervised experiments on NER to

Lang.	Method	1k	2k	4k	average
en	gold×1	59.74	69.14	76.48	68.45
	gold×2	60.92	69.79	76.57	69.09
	gold×3	61.13	70.42	74.92	67.24
	gold×4	61.15	**70.61**	**76.82**	**69.53**
	gold×5	**61.43**	70.38	76.43	69.41

Table 11: CoNLL NER F1: comparison on different oversampling ratios.

generate synthetic data, train the sequence tagging models and evaluate on the POS test data.

Results and Analysis We report accuracy of **wt** and **gen$_{ud}$** (average of 3 runs) in Table 12. Our method outperforms the baseline method **wt** when the number of gold sentences are less than 8k. When the number of gold sentences are more than 8k, the performance of our method is comparable with **wt**.

Method	1k	2k	4k	6k	8k	all
gold	79.18	82.17	85.83	88.62	90.21	93.00
+wt	81.11	84.00	86.91	89.64	**90.88**	**93.20**
+gen$_{ud}$	**82.11**	**84.93**	**87.52**	**89.98**	90.84	93.12

Table 12: Semi-supervised POS accuracy.

A.5 Synthetic Data Diversity: Unique Entities

To quantitatively measure the diversity introduced by our method in the supervised English NER tasks, we count the number of unique entities (without context) in the gold and generated data. Results are presented in Figure 6.

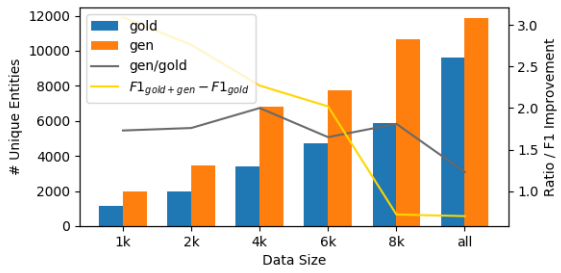

Figure 6: Statistics of unique entities (without context)

A.6 Average Runtime

Table 13 is an illustration of the average runtime of our models in English NER, POS and E2E-TBSA tasks and RNNLM.

Task	1k	2k	4k	6k	8k	all
NER	26.5	70.5	124.4	167.9	216.3	393.2
POS	83.6	112.3	231.1	257.7	277.2	298.0
E2E-TBSA	-	89.3	150.8	269.2	-	-
RNNLM	0.7	1.1	2.0	2.7	3.3	3.9

Table 13: Average runtime (min).

A.7 Computing Infrastructure

We conduct our experiments on NVIDIA V100 GPU.