World Wide Web
https://doi.org/10.1007/s11280-021-00978-0

Description and demonstration guided data augmentation for sequence tagging

Zhuang Chen[1] · **Tieyun Qian**[1]

Received: 19 September 2021 / Revised: 8 November 2021 / Accepted: 12 November 2021 /
Published online: 11 December 2021

Abstract
Fine-grained annotations are indispensable for sequence tagging tasks like named entity recognition and aspect-based sentiment analysis, which may incur extremely high time and labor costs. Recent efforts are towards data augmentation which aims to generate synthetic labeled instances. However, most existing methods adopt the random replacement or perturbation strategy under pre-defined constraints, and thus often lead to unstable performance. More importantly, these methods focus on producing more artificial samples yet neglect to make good use of real training samples. In this paper, we propose a novel description and demonstration guided data augmentation (D^3A) approach for sequence tagging. On one hand, we collect *dependency paths as descriptions* to supervise the instance-level augmentation process, such that we can consistently generate high-quality synthetic data. On the other hand, we retrieve *semantic or syntactic related features as demonstrations* to enhance the learning capability of neural networks under limited training data. We conduct extensive experiments on four sequence tagging datasets with various sizes of training data. The results demonstrate that our proposed D^3A approach can significantly improve the performance of sequence tagging, especially in low-resource scenarios.

Keywords Data augmentation · Sequence tagging · Named entity recognition · Aspect-based sentiment analysis

1 Introduction

Deep learning methods usually need large-scaled labeled data to train neural networks. Considering that collecting and annotating data induce high time and labor costs, recent studies have paid attention to data augmentation techniques for automatically generating synthetic

✉ Tieyun Qian
qty@whu.edu.cn

Zhuang Chen
zhchen18@whu.edu.cn

[1] School of Computer Science, Wuhan University,
16 Luojiashan Road, Wuhan, 430072, Hubei, China

instances and increasing the data diversity [38]. Data augmentation is firstly used in the field of computer vision (CV), where photos are augmented by rotation, cropping, masking, color jittering, gray scaling, etc [15, 41]. Then data augmentation is quickly extended to the field of natural language processing (NLP). Different from photos in CV, languages in NLP are more sophisticated since even a slight modification may change the original semantic meanings. To avoid changing the labels after perturbing languages, existing data augmentation studies in NLP mainly concentrate on coarse-grained sentence-level tasks such as machine translation [8, 37], text classification [14, 44], and question answering [1, 21]. The frequently used methods for perturbing languages include back translation, random insertion, random swap, random deletion, etc.

Data augmentation research on fine-grained sequence tagging tasks like name entity recognition (NER) and aspect-based sentiment analysis (ABSA) is still limited. The main reason is that sequence tagging tasks are defined at the token level and neural models are trained to capture the one-to-one correspondence between tokens and their labels, and perturbing tokens sequences may produce the wrong label sequences.

For example, if "*Disney*" is deleted from the text segment "*love Disney Land*" tagged as "`O B-facility I-facility`", the perturbed label sequence will be "`O I-facility`" which is not allowed by the tagging rule in NER (no beginning of an entity).

Among the few data augmentation methods for sequence tagging tasks, most of them modify sentence-level perturbation methods by adding additional constraints to keep the correspondence of labels [4]. Related techniques include label-wise token replacement, shuffle within segments, mention replacement, etc. The main drawback of this type of methods is that they are not stable enough to obtain high-quality synthetic data due to the randomness in perturbation. Another type of methods generates synthetic instances by pre-training a customized language model and then sampling from it [7]. However, the language model needs to be trained on enough labeled data, which is not suitable for low-resource scenarios. Moreover, when sampling synthetic instances from the language model, several manual rules must be defined in advance and then be used to filter out low-quality outputs. Besides the above specific drawbacks, these two types of methods are both limited to the instance-level augmentation. In other words, they only focus on how to generate more synthetic instances, but neglect to promote sequence taggers[1] under the limited training data.

In this paper, we propose a description and demonstration guided data augmentation (D^3A) method for sequence tagging, which not only enhances the quality of the produced synthetic instances, but also generalizes the learning capability of the neural models. In particular, the description is a collection of dependency paths that act as the reference for producing new data for instance-level augmentation, and the demonstration is a set of syntactic or semantic related tokens that serve as the evidence for feature-level augmentation.

At the instance level, our goal is to generate more reliable synthetic instances with the help of *descriptions*. In the sequence tagging task, we can divide the token sequence into two types of tokens, namely mentions (e.g., named entities in NER and aspect terms in ABSA) and contexts (non-mention tokens). To increase the data diversity, mention replacement is a feasible way that keeps a mention's contexts unchanged and replaces itself with another mention to create a synthetic instance. However, existing methods [4] often select mentions for replacement at random and thus cannot ensure the high quality of synthetic instances. To

[1] In this paper, the sequence tagger denotes a specific neural network for sequence tagging.

solve this problem, we propose to construct descriptions for mentions to refine the replacement procedure. Specifically, we first summarize each mention with a description set that includes the involved dependency paths (e.g., $\texttt{MENTION} \xrightarrow{nsubj} \texttt{JJ}$). Then, in order to find the compatible mentions that match the original contexts, we calculate the correlation between the original mention and each candidate substitutive mention. Lastly, we rank the candidate mentions w.r.t their correlations and randomly choose several mentions that are qualified for synthesizing new instances.

At the feature level, we turn to make better use of the real but limited training sample. For this purpose, we introduce *demonstrations* for the token sequence to enhance the learning capability of sequence taggers. Specifically, for a token in the sequence, its demonstrations consist of a set of tokens that have appeared in training instances. These demonstration tokens play similar syntactic and semantic roles as the original token and could provide extra evidence for tagging. After augmenting token features with demonstrations in the learning procedure, a sequence tagger can model an instance not only under the guidance of its own label, but also under the guidance of other training instances. Consequently, the coupling relationship among instances can help the sequence tagger converge to a better state than before.

We conduct extensive experiments on two sequence tagging tasks including NER and ABSA, with two datasets per task. The experimental results demonstrate that after introducing descriptions at the instance level and demonstrations at the feature level, our proposed method can significantly improve the performance of sequence taggers, especially in low-resource scenarios.

2 Related work

In this section, we first introduce two related sequence tagging tasks: NER and ABSA. We then present a brief summarization of existing data augmentation methods towards sequence tagging tasks.

2.1 Named entity recognition (NER)

NER is a task for identifying and categorizing target keyphrases (entities) such as person, organization, time, and location. NER usually acts as the first step in the NLP processing pipeline, and serves as the information source for downstream tasks like question answering, information retrieval, and relation extraction. Early studies mainly use handcrafted features, manual rules, and linguistic lexicons [22, 30, 35]. Recent studies focus on designing neural models which require little feature engineering and expert knowledge [16, 19, 46, 48, 53]. Under the deep learning framework, NER is typically defined as a sequence tagging task. For example, in the text sentence "*Made it back home to GA. Time to start planning the next Disney World trip.*", "GA, Disney World" are tagged as "`B-Location, B-facility I-facility`", respectively, while other tokens are tagged as "O".[2]

[2] In this paper, we use the B(beginning)-I(inside)-O(outside) tagging scheme throughout. Other schemes such as B-I-O-E(end)-S(single) can also be used as labels. The choice of tagging scheme does not affect the implementation of our method.

2.2 Aspect-based sentiment analysis (ABSA)

ABSA is a fine-grained task that aims to summarize the opinions of users towards specific aspects in reviews. With the rapid growth of the world wide web and social media, ABSA has been widely applied to various fields to analyze texts like product reviews, forum discussions, and blog posts. For example, given a sentence "The pizza here is also absolutely delicious.", ABSA needs to extract the aspect term and classify the corresponding sentiment polarity, then tag "pizza" with "`B-positive`" and other tokens with "`O`". Most existing studies treat ABSA as a two-step task and develop separate methods for aspect term extraction [23, 24, 26, 34, 42, 47, 51] and aspect-level sentiment classification [2, 11, 12, 18, 20, 25, 54]. To obtain the complete ABSA predictions, results from two steps must be merged together in a pipeline manner, which however may lead to error propagation. To address this problem, recent studies in ABSA design end-to-end sequence taggers that directly map tokens to their collapsed labels [3, 17, 28, 50].

2.3 Data augmentation for sequence tagging

Data augmentation are originated from computer vision [38] and then quickly applied to natural language processing [9]. Most existing data augmentation methods are designed for document-level and sentence-level tasks such as machine translation [8, 37], text classification [14, 44], and question answering [1, 21]. Frequently used methods for perturbing languages include back propagation, random insertion, random swap, random deletion, etc. In this scenario, synthetic data is relatively easy to obtain since the labels usually remain unchanged after language perturbation.

For fine-grained sequence tagging tasks, where tokens and labels have a fragile one-to-one correspondence, there are only a few studies available. Sahin and Steedman [36] use dependency tree morphing (sentence cropping and sentence rotating) to generate synthetic data for POS tagging, but the produced synthetic data is not semantically smooth and hard to interpret. Ding et al. [7] first pre-train a customized language model by concatenating tokens with their labels, then sample outputs from this language model and transform them to the synthetic data. Their language model is trained on at least $1k$ annotated sentences, which is not suitable for low-resource scenarios. Moreover, several manual rules need to be pre-defined and then used to filter out low-quality outputs. Dai et al. [4] modify sentence-level augmentation methods by adding additional constraints, and propose several methods for NER including the label-wise token replacement, synonym replacement, and mention replacement. These methods show improved performance in both recurrent and transformer models, but the synthetic data is not stable enough and contains harmful noises. Zhang et al. [52] adapt the MIXUP [49] technique for active sequence labeling by augmenting queried samples which also requires manual labor. Guo et al. [10] also refer to MIXUP and create new synthetic instances by softly combining token/label sequences following the Beta distribution, which blends mentions with non-mentions and does not perform well for sequence tagging. Apart from the specific drawbacks, existing methods for data augmentation are all limited to instance-level augmentation. They only pay attention to synthesize instances, but neglect to promote taggers to make better use of the limited training data.

Different from prior methods, our proposed approach generates synthetic samples with the guidance of descriptions, and hence it can produce stable results. In addition, our approach performs feature-level augmentation for training samples, which can further improve the learning capability of neural networks with limited training data.

3 Methodology

In this section, we first introduce the definition of sequence tagging and the overview of the proposed method. We then present two backbone sequence taggers as the carrier and tester of data augmentation methods. Lastly, we illustrate our method in detail.

3.1 Problem definition

In sequence tagging tasks, a sequence tagger is trained to learn a mapping function f between a token sequence $x = \{x_1, ..., x_n\}$ and a label sequence $y = \{y_1, ..., y_n\}$, i.e., $f : x \rightarrow y$. Each x_i is a token in natural language and each y_i belongs to the label set {`B-X`, `I-X`, `O`}. Specifically, the first token of a mention with the type `X` is tagged as `B-X`, and the tokens inside that mention are tagged as `I-X` and the contexts (non-mention tokens) are tagged as `O`.

In this work, we focus on the data augmentation issue for sequence tagging. Given a training set containing (usually limited) gold-labeled instances $\mathcal{D}^{train}$, our goal is to generate some synthetic labeled instances $\mathcal{D}^{syn}$ to improve the diversity of the training set, and promote the sequence tagger's performance in tagging the unseen test set $\mathcal{D}^{test}$.

3.2 Backbone sequence tagger

To examine the effectiveness of different data augmentation methods, we consider two types of backbone sequence taggers: one is based on static GloVe embeddings, and the other is based on contextual BERT representations.

In the GloVe backbone, we first map each token in x_i with the static GloVe embeddings and obtain its vector e_i, then use an additional encoder containing several convolutional layers without pooling operation to extract the hidden state h_i:

$$\begin{aligned} \{e_1, ..., e_n\} &= \textit{GloVe-Lookup}\ (\{x_1, ..., x_n\}), \\ \{h_1, ..., h_n\} &= \textit{CNN-Encoder}\ (\{e_1, ..., e_n\}), \end{aligned} \tag{1}$$

where the parameters of the CNN encoder are learned from scatch. On the contrary, in the BERT backbone, we directly use the pre-trained BERT encoders to obtain the hidden state h_i:

$$\{h_1, ..., h_n\} = \textit{BERT-Encoder}\ (\{x_1, ..., x_n\}) \tag{2}$$

In both backbones, after extracting the hidden state of each token, a classifier which consists of a linear transformation layer and a softmax function is used to predict the tags of tokens:

$$\{\hat{y}_1, ..., \hat{y}_n\} = \textit{Classifier}\ (\{h_1, ..., h_n\}) \tag{3}$$

Lastly, we compute the cross-entropy loss and train learnable parameters with back propagation:

$$\mathcal{L} = -\sum_{i=1}^{n} \sum_{j=1}^{J} \hat{y_{ij}} \cdot log(y_{ij}), \tag{4}$$

where n is the length of x, J is the number of label categories, $\hat{y_i}$ and y_i are the predictions and ground truth labels, respectively. Given the gold instances from $\mathcal{D}^{train}$ and the synthetic instances from $\mathcal{D}^{syn}$, we can train the backbone sequence taggers accordingly and then make inference on $\mathcal{D}^{test}$.

3.3 Description guided instance-level augmentation

At the instance level, D^{3}A aims to synthesize reliable instances and then adds them into the training set for boosting the performance and generalization of sequence taggers. For example, for a token sequence "*Can't upload payload to my apache 2 server, pentesting exercise.*" in the NER task, we can divide it into two types of tokens, namely the mention (i.e., "apache 2 server") and the contexts (non-mention tokens). For improving the diversity of the training data, Dai et al. [4] propose to keep a mention's contexts unchanged and replace the mention with another one to generate a synthetic instance. However, they select the substitutive mentions only based on the mention type (e.g., "PRODUCT" in this example). Consequently, an incompatible mention like "Touchscreen" may be selected to join the contexts and cause incongruity in semantics. Different from this naive mention replacement method, we propose to construct descriptions for mentions by resorting to the involved dependency paths and then search compatible substitutive mentions for replacement. In this way, our proposed D^{3}A can ensure the quality of the synthetic data and can conduct effective instance-level data augmentation.

3.3.1 Construct descriptions via dependency paths

To capture the correlation among different mentions, we need to characterize mentions completely. Therefore, we propose to construct descriptions for mentions according to their involved dependency paths. As shown in Figure 1, for the NER instance with the mention "apache 2 server" and the mention type "PRODUCT", we can construct the mention's description set by collecting the involved dependency paths. According to the parsing results, the paths can be divided into two categories as follows.

- **Head-related paths**. In a dependency parse tree, each token has exact one head token (a.k.a, governor). Inspired by a recent study for aspect-level sentiment classification [43], we first reshape the original parsing tree to a mention-oriented tree so as to keep the integrity of the target mention. Specifically, we treat the mention containing multiple tokens as a whole and only consider the paths outside the mention. For example, two paths inside the target mention, i.e., apache $\xrightarrow{nummod}$ 2 and

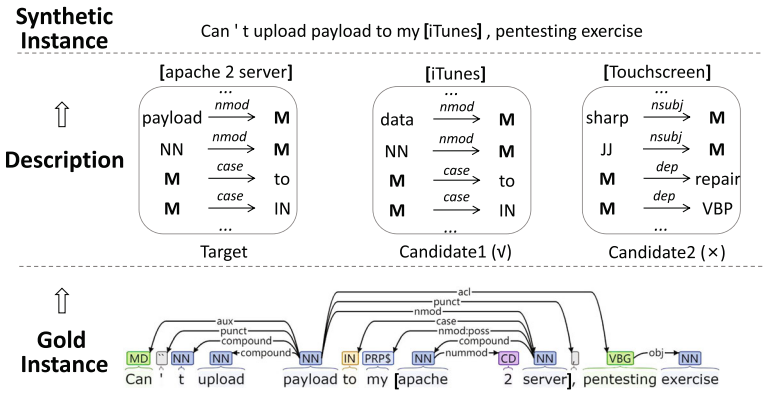


Figure 1 Illustration of the description guided instance-level data augmentation

$\texttt{server} \xrightarrow{compound} \texttt{apache}$, are discarded. After reshaping, we can easily collect two head-related paths: $\texttt{payload} \xrightarrow{nmod} \texttt{M}$ and $\texttt{NN} \xrightarrow{nmod} \texttt{M}$ that corresponds the head token and its part-of-speech (POS) tag, respectively. (Here we use M to denote an arbitrary mention for simplicity.)

- **Tail-related paths**. Similar to the head-related paths, after reshaping the parsing tree, we can collect four tail-related paths: $\texttt{M} \xrightarrow{case} \texttt{to}$, $\texttt{M} \xrightarrow{case} \texttt{IN}$, $\texttt{M} \xrightarrow{nmod:pass} \texttt{my}$, and $\texttt{M} \xrightarrow{nmod:pass} \texttt{PRP\$}$. One small difference here is that the number of tail tokens (a.k.a, dependent) is not limited to only one.

After traversing the entire training data, for each mention M, we can construct its description (the set S) that contains all involved dependency paths.

3.3.2 Search compatible substitutive mentions

For the target mention M, we consider all other mentions that belong to the same type as its candidate substitutive mentions. Once the descriptions are constructed, we start to search compatible candidates to replace the target mention and generate synthetic instances. Given the target mention M and a random candidate mention $\hat{\texttt{M}}$, we calculate their correlation via the Jaccard similarity of their corresponding description sets S and $\hat{\texttt{S}}$:

$$Correlation(\texttt{M}, \hat{\texttt{M}}) = Jaccard(\texttt{S}, \hat{\texttt{S}}) = \frac{|\texttt{S} \cap \hat{\texttt{S}}|}{|\texttt{S} \cup \hat{\texttt{S}}|}. \tag{5}$$

By ranking candidate mentions based on the Jaccard similarities, we can preserve the top $\tau\%$ related candidates and filter out the others, where τ is a hyperparameter. Generally, τ is inversely proportional to the size of the training data $\mathcal{D}^{train}$ as we will show in the analysis section. After that, we randomly select a substitutive mention $\hat{\texttt{M}}$ from the preserved candidates, then fix $\hat{\texttt{M}}$ and the contexts of M to generate a synthetic instance. By this means, we refine the selection process in the naive mention replacement method and obtain more reliable synthetic instances $\mathcal{D}^{syn}$. Finally, the instance-level data augmentation can be achieved by training backbone sequence taggers with the combination of $\mathcal{D}^{train}$ and $\mathcal{D}^{syn}$.

3.4 Demonstration guided feature-level augmentation

Existing data augmentation methods for sequence tagging tasks focus on synthesizing new instances when the labeled training set is small. However, we can also consider this issue from another point of view, i.e., how to make full use of limited labeled data. To this end, we propose to further conduct feature-level data augmentation by enhancing the training process with demonstrations. Specifically, when extracting features from the token sequence, the sequence tagger will receive a set of demonstration tokens that can provide extra information for tagging. As shown in Figure 2, for a rare token "apache" without enough sample exposure, the tagger can hardly predict its correct label "B-PRODUCT". However, if we associate "apache" with some demonstrations like "proxy, git, anaconda, android", the tagging of "apache" will become less challenging.

We now illustrate the feature-level data augmentation in two steps, i.e., retrieving demonstrations from the training data and augmenting token features with demonstrations. We take a training instance $x = \{x_1, ..., x_n\}$ in $\mathcal{D}^{train}$ as the example.

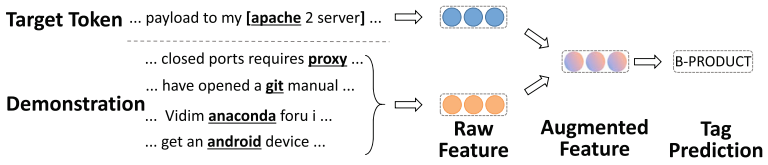


Figure 2 Illustration of the demonstration guided feature-level data augmentation

3.4.1 Retrieve demonstrations from training data

For a target token $x_i \in x$, we define its demonstrations as a set of tokens $\{d_1, d_2, ..., d_K\}$ where each d_j has similar syntactic and semantic characteristics with x_i. For convenience, we correspond both x_i and d_j to the word vocabulary $\mathcal{V}$ and change the notations accordingly. Then the problem can be reformulated as follows: for a target token v, how to select another token $\widetilde{v} \in \mathcal{V}^{train}$ (the vocabulary of $\mathcal{D}^{train}$) that is qualified to serve as v's demonstration. To answer the problem, we resort to three different attributes: the semantic meaning, the part-of-speech tag, and the dependency relation.

- **Semantic meaning**. We use the pre-trained GloVe embedding to obtain the vectors v_{sem} and $\widetilde{v}_{sem}$ for v and $\widetilde{v}$, respectively. We then calculate the semantic similarity between v and $\widetilde{v}$:

$$sem.sim(v, \widetilde{v}) = cosine(v_{sem}, \widetilde{v}_{sem}), \tag{6}$$

  where $cosine(\cdot, \cdot)$ is the cosine similarity.
- **Part-of-speech tag**. In each sentence where v has appeared, we can use a one-hot vector $v_{pos} \in \mathcal{R}^{N_{pos}}$ to represent its POS tag, where N_{pos} is the number of POS types. Notice that many tokens (e.g., "like") are polysemous and can serve as different POS tags in different contexts. Therefore, we choose to summarize the global usages $< v_{pos} >$ of v by merging its POS vectors in all sentences:

$$< v_{pos} >= \{v_{pos,l=1} \mid v_{pos,l=2} \mid ... \mid v_{pos,l=|\mathcal{D}^{train}|}\} \tag{7}$$

  where $\mid$ is the dimension-wise OR operation. Similarly, we can obtain $< \widetilde{v}_{pos} >$ for $\widetilde{v}$:

$$< \widetilde{v}_{pos} >= \{\widetilde{v}_{pos,l=1} \mid \{\widetilde{v}_{pos,l=2} \mid ... \mid \{\widetilde{v}_{pos,l=|\mathcal{D}^{train}|}\} \tag{8}$$

  We then calculate the POS similarity between v and $\widetilde{v}$ as follows:

$$pos.sim(v, \widetilde{v}) = cosine(< v_{pos} >, < \widetilde{v}_{pos} >). \tag{9}$$

- **Dependency relation**. As we illustrated in Section 3.3, dependency relations can be divided into head- and tail-related ones. In each sentence where v has appeared, we can use a one-hot vector v_{head} and a multi-hot vector v_{tail} to represent the involved head and tail relation, where each vector $\in \mathcal{R}^{N_{dep}}$ and N_{dep} is the number of relation types. Then we concatenate them to form the whole dependency vector $v_{dep} \in \mathcal{R}^{2 \times N_{dep}}$. Following the steps in calculating the POS similarity, we can obtain the global usages $< v_{dep} >$ for v and $< \widetilde{v}_{dep} >$ for $\widetilde{v}$, then calculate the dependency similarity between v and $\widetilde{v}$:

$$dep.sim(v, \widetilde{v}) = cosine(< v_{dep} >, < \widetilde{v}_{dep} >). \tag{10}$$

After calculating three different types of attributes' similarities, we can obtain the overall similarity score between v and $\widetilde{v}$:

$$attr.sim(v, \widetilde{v}) = sem.sim + pos.sim + dep.sim, \tag{11}$$

Consequently, we can obtain a $attr.sim$ score matrix $\mathbf{M}^{train} \in \mathcal{R}^{|V^{train}|\times|V^{train}|}$. After ranking, for v, we select the top-K tokens as the demonstrations $\{d_1, d_2, ..., d_K\}$, and record their $attr.sim$ scores $\{a_1, a_2, ..., a_K\}$.

We retrieve demonstrations for both training and test data, so they can benefit both the training and inference processes. During testing, we can calculate the *attr.sim* score matrix $\mathbf{M}^{test} \in \mathcal{R}^{|V^{test}|\times|V^{train}|}$ in a similar way. However, a problem here is that we cannot obtain $< v_{pos} >$ and $< v_{dep} >$ since the whole test data is unseen. Therefore, we use the local v_{pos} and v_{dep} of the current test sample to calculate the part-of-speech and dependency similarities. The retrieval process is a one-time job and often finishes in ten seconds.

3.4.2 Augment token features with demonstrations

After obtaining demonstrations, we can conduct the feature-level augmentation augment with these demonstrations. Generally, we follow a simple rule, i.e., injecting the demonstrations after the pre-trained module. Moreover, considering the difference in the amount of information carried by GloVe and BERT, we propose two augmentation methods accordingly.

In the GloVe backbone, our target is the vector e_i of each token x_i. Specifically, we first map x_i's demonstration tokens $d_{i,k}$ to the vectors $d_{i,k}$, then aggregate them to a single vector $\widetilde{d}_i$ according to the similarity scores $a_{i,k}$:

$$\begin{aligned} \{d_{i,1}, ..., d_{i,K}\} &= \textit{GloVe-Lookup}\ (\{d_{i,1}, ..., d_{i,K}\}), \\ \widetilde{d}_i &= \sum_{k=1}^{K} d_{i,k} \cdot a_{i,k}. \end{aligned} \tag{12}$$

We then calculate a dimension-wise gate g_i to augment e_i with $\widetilde{d}_i$:

$$\begin{aligned} g_i &= \sigma\ (\mathbf{W}_1(e_i \oplus \widetilde{d}_i)), \\ r_i &= g_i \odot (e_i \oplus \widetilde{d}_i), \end{aligned} \tag{13}$$

where e_i is the GloVe word vector, $\widetilde{d}_i$ is the demonstration vector, $\mathbf{W}_1$ is a transformation matrix, σ is the Sigmoid function, $\oplus$ is concatenation, and $\odot$ is element-wise multiplication. Lastly, we send the augmented vector r_i instead of e_i to the blank CNN encoder for extracting hidden states while other modules remain unchanged.

In the BERT backbone, we adopt a different strategy since the contextualized BERT representations are very informative. Therefore, we do not interfere the encoding process but inject demonstrations into the hidden states h_i. Specifically, we first use the embedding layer inside BERT to transform $d_{i,k}$ into $\widetilde{d}_i$, then pass them to the BERT encoder and obtain the hidden states $\widetilde{h}_i$. Afterwards, we calculate a single-value gate g_i to combine h_i and $\widetilde{h}_i$:

$$\begin{aligned} g_i &= \sigma\ (\mathbf{W}_2(h_i \oplus \widetilde{h}_i)), \\ r_i &= g_i \cdot h_i + (1 - g_i) \cdot \widetilde{h}_i, \end{aligned} \tag{14}$$

where $\mathbf{W}_2$ is a transformation matrix, h_i and $\widetilde{h}_i$ are the hidden states of input tokens and demonstrations. Lastly, we send r_i instead of h_i to the token classifier and make predictions. After augmenting token features with demonstrations in the learning procedure, a sequence tagger can model an instance not only under the guidance of its own label, but also under the guidance of other training instances. Consequently, the coupling relationship among instances can help the sequence tagger converge to a better state given limited training data.

4 Experiment

In this section, we first present the experimental setup, then compare the proposed D^3A method with the state-of-the-art data augmentation baselines.

4.1 Experimental setup

4.1.1 Datasets

We examine D^3A on two sequence tagging tasks: named entity recognition (NER) and aspect-based sentiment analysis (ABSA), each containing two datasets. For NER, we use the WNUT16 [40] and WNUT17 [5] constructed from Twitter and adopt the original train/test/development splits for the experiment. For ABSA, we merge the restaurant datasets from the ABSA tasks in SemEval 2014 [33], 2015 [32], and 2016 [31], and the laptop dataset from SemEval 2014 Task 4 [33]. Since there are no official development data, we randomly sample 20% training instances from each dataset as the development set, and use the rest instances for training. The detailed statistics of datasets are presented in Table 1. We use four different sizes of datasets to examine the effectiveness of data augmentation methods in different scenarios. `SMALL (S)` contains 50 training instances, `MEDIUM (M)` contains 150 training instances, `LARGE (L)` contains 300 training instances, and `FULL (F)` uses the complete training set. Generally, `S`, `M`, and `L` settings can be considered as the low-resource scenarios.

4.1.2 Settings

We pre-process each dataset by lowercasing all words and use Stanford CoreNLP [27] for dependency parsing. There are N_{pos}=45 classes of POS tags and N_{dep}=40 classes of dependency relations in four datasets.

In the GloVe-based backbone, we use the `glove.840B.300d.txt` vectors. The kernel size and the number of convolution layers in the CNN encoder are set to 3 and 4, respectively. Dropout [39] is applied to convolution layers' outputs with the probability of

Table 1 The statistics of datasets

Task	Dataset	Split	Sentences	Tokens	Mentions	Types
ABSA	Restaurant	Train	3102	48094	3433	4
		Dev	775	11875	881	4
		Test	2158	32639	2289	4
	Laptop	Train	2436	40748	1830	4
		Dev	609	10027	473	4
		Test	800	11699	634	4
NER	WNUT16	Train	2394	49004	1496	11
		Dev	1000	17698	661	11
		Test	3850	68015	3473	11
	WNUT17	Train	3394	66702	1975	7
		Dev	1008	15785	835	7
		Test	1287	24471	1079	7

0.5. In the BERT-based backbone, we use the officially released `bert-base-uncased` pre-trained model [6]. We train the GloVe/BERT backbone for 100/15 epochs using Adam optimizer [13] with the learning rate 1e-4/3e-5 and batch size 8 in a 3090 GPU, respectively.

In the description guided instance-level augmentation, the threshold τ for preserving candidate mentions are tuned from 0.1 to 1.0, stepped by 0.1. In the demonstration guided feature-level augmentation, we set the number of demonstrations K=10. If there are synthetic instances in the training data, feature-level augmentation will also be conducted on these instances.

4.1.3 Evaluation protocol

We report F1-scores for both NER and ABSA tasks in different scenarios, and also present the mean improvement δ for clear comparison. To compute F1 scores, the prediction would be considered correct if it exactly matches the mention span and mention type. We run the experiments five times with random initialization and report the averaged results. The checkpoint achieving the maximum F1-score on the development set is used for evaluation on the test set.

4.2 Compared methods

Details of compared methods are listed below. For all data augmentation methods, the ratio of gold data to synthetic data is 1:3, which means the augmented training set is four times larger than before.

- **NoAug** : No augmentation. It only uses the gold training data.
- **DUP** : Duplication. A naive augmentation method that simply duplicates the gold data three times. This is an important baseline to observe the effectiveness of other data augmentation methods.
- **LwTR** : Label-wise token representation [4]. For each token in the sequence, a binomial distribution is used to randomly decide whether it should be replaced. If yes, the token is replaced by a randomly selected token with the same label.
- **SR** : Synonym replacement [4]. It is similar to LwTR, except that the token is replaced with one of its synonyms retrieved from WordNet.
- **MR** : Mention replacement [4]. For each mention in the instance, a binomial distribution is used to randomly decide whether it should be replaced. If yes, the mention is replaced by another mention from the original training set which has the same entity type.
- **SiS** : Shuffle within segments [4]. It first splits the token sequence into segments of the same label and makes each segment corresponds to either a mention or a sequence of out-of-mention tokens. Then for each segment, a binomial distribution is used to randomly decide whether it should be shuffled. If yes, the order of the tokens within the segment is shuffled, while the label order is kept unchanged.
- **SeqMix** : Sequence mixup [10]. It creates new synthetic instances by softly combining token/label sequences from the training data. The proportion of the mixture is sampled from a Beta distribution.
- **DAGA** : Data augmentation with a generation approach [7]. It is a two-step augmentation method. First, a language model over sequences of labels and words linearized as per a certain scheme is learned. Second, sequences are sampled from the fixed language model and de-linearized to generate new tokens and labels.

4.3 Main results

The comparison results of different methods are shown in Table 2. Clearly, our proposed D^3A achieves a new state-of-the-art performance on all four datasets. For GloVe-ABSA, BERT-ABSA, GloVe-NER, and BERT-NER,[3] D^3A outperforms NoAug by 11.94%, 5.22%, 7.40%, and 6.77%. It also outperforms the second-best augmentation baselines by 2.90%, 1.75%, 0.66%, and 1.60%, respectively. When inspecting the results in detail, we can further draw three conclusions.

Firstly, compared with the backbone models without synthetic data (i.e., NoAug in each table), almost all data augmentation methods can improve the performance in sequence tagging tasks. However, we found that simply duplicating gold data several times (i.e., DUP) can also achieve promising performance, and even surpass some baseline methods like LwTR occasionally. The reason is that, in NoAug, the hyper-parameters of sequence taggers are identical to other methods but the training instances in settings `S`, `M`, and `L` are very limited. Therefore, the insufficient exposure of instances causes the underfitting of sequence taggers and deteriorates the performance. After duplicating training instances, the taggers can converge to a better state than before and achieve a performance gain. We believe that the comparison with DUP is an important standard for judging the effectiveness of data augmentation methods, but it is often ignored by previous work. Obviously, the proposed D^3A consistently outperforms DUP on the mean δ of F1-scores in all scenarios.

Secondly, compared with NoAug, the performance gain brought by data augmentation methods is approximately inversely proportional to the size of training data. In small (`S`) training sets, all data augmentation methods achieve significant improvements over NoAug. While in full (`F`) training sets, the performance becomes stable and even decreases in some scenarios. The reason is intuitive. When the training instances are inadequate, the mentions only co-occur with limited contexts. Therefore, synthetic instances can increase the data diversity and brings about performance gains. As the training set grows, more and more collocations between mentions and contexts are already covered by the gold data. On the contrary, some low-quality synthetic instances act as noise and even poison the sequence taggers. Therefore, the data augmentation methods sometimes become optional when there is enough training data.

Thirdly, augmenting BERT-based sequence taggers is more difficult than augmenting GloVe-based ones. For example, in the ABSA task, D^3A improves GloVe backbone by 11.94%, but this value for BERT backbone is only 5.22%. As shown in previous studies [4, 6], the stacked transformer encoders pre-trained on large-scale external data make BERT more powerful than static word embeddings like GloVe in natural language understanding. Therefore, compared with GloVe backbones, the knowledge carried by the synthetic instances is less useful for BERT backbones.

5 Deep analysis

In this section, we present an in-depth analysis including the augmentation of SOTA methods with D^3A, ablation study, parameter study, and case study.

[3]For simplicity, we here use Backbone-Task (e.g., GloVe-ABSA) pairs for illustration.

Table 2 Comparison of different methods for two sequence tagging tasks

(a) Comparison results of aspect-based sentiment analysis (ABSA)

Tagger	Method	Restaurant				Laptop				Mean δ
		S	M	L	F	S	M	L	F	
GloVe	**NoAug**	**26.56**	**36.32**	**39.87**	**62.36**	**5.49**	**14.66**	**25.16**	**50.33**	**/**
	DUP	33.70	42.87	46.86	67.00	21.95	29.19	35.31	51.43	8.45
	LwTR	31.21	40.93	44.98	61.81	21.16	25.87	33.35	48.19	5.84
	SR	33.01	42.19	46.12	63.94	20.48	26.28	32.81	49.01	6.64
	MR	32.93	43.19	47.49	65.75	23.40	30.28	36.81	53.20	9.04
	SiS	32.97	43.23	46.41	64.95	22.69	29.23	35.03	52.08	8.23
	SeqMix	30.44	40.73	44.40	63.05	22.08	26.49	34.59	50.42	6.43
	DAGA	31.62	42.07	44.99	63.50	26.62	31.14	36.03	52.56	8.47
	D^3A	**35.95**	**45.48**	**48.25**	**67.63**	**28.06**	**34.41**	**41.25**	**55.22**	**11.94**†
BERT	**NoAug**	**32.79**	**44.68**	**48.15**	**72.19**	**22.07**	**36.94**	**46.73**	**60.92**	**/**
	DUP	35.68	46.62	50.05	70.77	26.95	35.99	45.72	62.10	1.18
	LwTR	32.83	41.32	46.76	70.94	26.05	34.85	38.97	57.14	− 1.95
	SR	37.25	45.85	51.62	72.20	29.13	37.19	45.89	61.86	2.07
	MR	36.70	46.55	51.58	72.06	32.91	42.35	**49.16**	60.89	3.47
	SiS	34.59	46.51	49.76	**72.55**	31.56	34.80	41.41	61.29	1.00
	SeqMix	33.44	41.95	49.04	69.53	29.37	40.49	47.89	59.25	0.81
	DAGA	35.27	44.71	50.86	70.65	34.64	41.03	48.08	57.36	2.27
	D^3A	**37.71**	**47.51**	**51.97**	71.63	**39.05**	**45.64**	48.58	**64.18**	**5.22**†

(b) Comparison results of named entity recognition (NER)

Tagger	Method	WNUT16				WNUT17				Mean δ
		S	M	L	F	S	M	L	F	
GloVe	**NoAug**	**5.95**	**15.15**	**20.01**	**35.44**	**3.51**	**14.14**	**23.78**	**30.58**	**/**
	DUP	14.44	21.31	**27.92**	37.29	13.10	22.31	27.04	33.10	5.99
	LwTR	13.52	20.92	27.49	38.06	16.26	23.14	25.87	33.63	6.29
	SR	13.66	20.26	27.45	36.95	14.16	22.97	28.33	32.16	5.92
	MR	12.38	20.36	26.91	36.21	14.87	21.78	27.06	34.37	5.67
	SiS	**14.75**	**22.84**	27.59	37.47	15.97	24.23	27.24	32.35	6.74
	SeqMix	11.00	19.23	25.82	36.47	14.52	19.72	25.04	31.42	4.33
	DAGA	11.33	20.36	27.59	38.20	13.06	22.86	27.09	32.23	5.52
	D^3A	13.23	22.32	27.70	**38.36**	**17.00**	**25.86**	**28.39**	**34.91**	**7.40**
BERT	**NoAug**	**0.76**	**22.67**	**30.12**	**43.10**	**1.45**	**22.66**	**30.72**	**41.86**	**/**
	DUP	17.98	27.12	31.75	**42.63**	15.71	25.58	32.16	41.74	5.17
	LwTR	16.15	26.48	29.85	38.38	14.80	25.28	**34.36**	37.36	3.67
	SR	19.25	26.72	30.81	42.58	12.71	29.06	28.80	38.62	4.40
	MR	18.61	27.73	32.76	41.14	12.02	24.63	33.25	39.63	4.55
	SiS	18.42	26.14	32.89	42.17	11.98	25.27	30.47	38.39	4.05
	SeqMix	17.10	26.83	30.08	37.96	15.66	26.92	32.46	36.67	3.79
	DAGA	16.25	25.23	29.31	37.71	17.50	28.30	33.73	39.09	4.22
	D^3A	**19.32**	**29.57**	**33.14**	41.09	**18.70**	**29.96**	33.73	**41.97**	**6.77**†

The results in blue are derived from the backbone taggers without data augmentation. The best scores after augmentation are in orange and the second best ones are underlined. All results are average scores of 5 runs with random initialization, and those with † are significantly better than the second-best methods ($p < 0.01$) based on one-tailed unpaired t-test

Table 3 Augmentation of SOTA methods in ABSA and NER with D^3A

(a) Comparison with the SOTA method (BERT-PT) for ABSA

Tagger	Method	Restaurant				Laptop				Mean δ
		S	M	L	F	S	M	L	F	
BERT-PT	NoAug	43.39	53.55	58.20	74.84	38.69	48.29	54.62	66.03	/
	D^3A	44.18	55.91	59.85	74.94	40.48	54.20	59.41	64.77	$2.02^{\dagger}$

(b) Comparison with the SOTA method (SANER) for NER

Tagger	Method	WNUT16				WNUT17				Mean δ
		S	M	L	F	S	M	L	F	
SANER	NoAug	0.32	14.93	29.17	48.22	0.18	3.14	30.52	46.08	/
	D^3A	14.73	23.53	34.33	49.02	6.31	23.59	37.44	48.60	$8.12^{\dagger}$

For the results of NoAug, we directly run the source codes of SOTA methods under different settings. The improvements are significant under $p < 0.01$ based on one-tailed unpaired t-test

5.1 Augmentation of SOTA methods with D^3A

To demonstrate the effectiveness of D^3A, we further examine it with state-of-the-art methods for aspect-based sentiment analysis and named entity recognition, respectively. According to the public leaderboards,[4] we select BERT-PT[5] [45] and SANER[6] [29] as the competitors and present the results in Table 3.

For ABSA, BERT-PT post-trains the BERT model with in-domain corpus from the large-scale Yelp and Amazon reviews. With full training data, BERT-PT achieves 74.84% and 66.03% F1-scores on the Restaurant and Laptop datasets (our best backbone achieves 72.19% and 60.92%). For NER, SANER trains the transformer encoders with semantic augmentation. With full training data, SANER achieves 48.22% and 46.08% F1-scores on the WNUT16 and WNUT17 datasets (our best backbone achieves 43.10% and 41.86%).

After augmenting SOTA methods with D^3A, we further obtain 2.02% and 8.12% mean improvements on two tasks. The improvements mainly come from situations with limited training data like `S` and `M`. Since BERT-PT is fully pre-trained, it is relatively stable under all settings. On the contrary, the encoders inside SANER are trained from scratch and can only achieve promising performance when training data is adequate.

5.2 Ablation study

To validate the effectiveness of designs in D^3A, we conduct a series of ablation study on both instance- and feature-level augmentation. The results are presented in Table 4.

In variants 1∼2, we remove the feature-level or instance-level augmentation respectively, and the drop of F1-scores demonstrates the effectiveness of both levels of augmentation. Moreover, the description guided instance-level augmentation is more important than the demonstration guided feature-level augmentation, especially in low-resource scenarios (`S`,

[4] https://paperswithcode.com/sota/aspect-based-sentiment-analysis-on-semeval-7 for ABSA and https://paperswithcode.com/sota/named-entity-recognition-on-wnut-2016 for NER.

[5] https://github.com/howardhsu/BERT-for-RRC-ABSA.

[6] https://github.com/cuhksz-nlp/SANER. Since the best method CL-KL uses external resources, we select the second-best one SANER. We do not include the development set for training.

Table 4 Ablation study

(a) Ablation study of aspect-based sentiment analysis (ABSA)

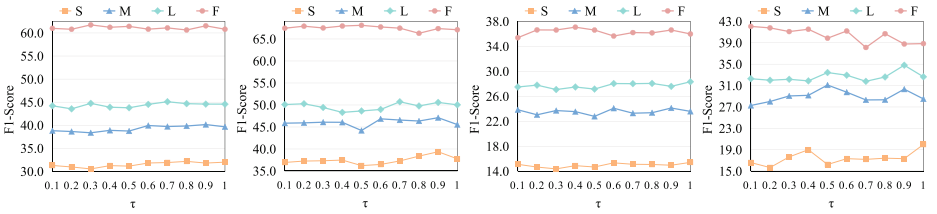
Tagger	Method	Restaurant				Laptop				Mean δ
		S	M	L	F	S	M	L	F	
GloVe	D^3A	35.95	45.48	48.25	67.63	28.02	34.41	41.25	55.22	**11.94**
	1.remove feature	33.96	44.05	47.79	65.73	23.31	30.82	37.56	53.20	9.46
	2.remove instance	31.78	41.01	44.46	64.82	10.16	15.88	33.33	55.73	4.55
	3.only head path(I)	35.88	44.74	48.32	67.31	26.56	33.80	39.91	55.85	11.45
	4.only tail path(I)	35.21	45.97	48.96	66.97	29.36	32.84	38.93	55.99	11.69
	5.only sem.sim(F)	35.23	45.18	48.49	67.30	27.96	34.25	39.06	55.90	11.58
	6.only pos.sim(F)	35.59	43.51	48.62	67.22	28.95	36.03	38.05	55.07	11.54
	7.only dep.sim(F)	35.47	46.05	48.27	67.18	27.04	34.94	40.88	55.91	<u>11.87</u>
BERT	D^3A	37.71	47.51	51.97	73.29	39.05	45.64	48.58	64.18	**5.22**
	1.remove feature	37.77	48.06	52.38	72.94	32.97	41.36	50.76	61.67	4.18
	2.remove instance	36.92	44.42	51.40	72.38	30.65	39.24	47.96	61.93	2.55
	3.only head path(I)	35.61	46.52	51.63	72.45	35.32	38.66	49.58	59.34	3.08
	4.only tail path(I)	37.28	46.66	51.92	72.86	37.05	42.92	50.73	61.28	<u>4.53</u>
	5.only sem.sim(F)	38.95	47.17	52.59	72.66	36.50	42.67	48.09	59.32	4.19
	6.only pos.sim(F)	37.30	46.45	52.76	73.44	32.58	40.45	49.17	60.76	3.55
	7.only dep.sim(F)	35.60	47.47	51.45	71.50	34.07	42.63	47.70	61.14	3.39

(b) Ablation study of named entity recognition (NER)

Tagger	Method	WNUT16				WNUT17				Mean δ
		S	M	L	F	S	M	L	F	
GloVe	D^3A	13.23	22.32	27.70	38.36	17.00	25.86	28.39	34.91	**7.40**
	1.remove feature	14.61	22.27	29.09	38.30	15.69	22.78	27.99	34.58	7.09
	2.remove instance	9.49	14.11	21.35	35.10	15.42	22.43	25.62	33.56	3.57
	3.only head path(I)	10.71	20.78	28.71	38.61	16.85	24.75	28.64	35.21	6.96
	4.only tail path(I)	12.46	20.75	26.69	37.87	16.71	24.42	28.26	35.02	6.70
	5.only sem.sim(F)	13.47	22.02	30.30	39.36	16.77	23.96	27.93	32.05	<u>7.16</u>
	6.only pos.sim(F)	10.17	20.49	27.84	40.16	15.42	24.49	27.98	34.48	6.56
	7.only dep.sim(F)	11.45	18.93	27.07	38.79	15.62	21.86	26.65	34.28	5.76
BERT	D^3A	19.32	29.57	33.14	41.09	18.70	29.96	33.73	41.97	**6.77**
	1.remove feature	19.21	28.01	32.56	42.29	15.71	30.50	33.46	39.85	6.03
	2.remove instance	0.76	24.06	31.12	43.47	2.00	19.57	29.50	40.00	−0.36
	3.only head path(I)	18.76	27.63	31.63	41.03	17.02	26.99	34.58	39.88	5.52
	4.only tail path(I)	18.13	27.80	31.97	39.90	16.05	26.90	33.61	37.72	4.84
	5.only sem.sim(F)	18.47	26.87	32.84	40.51	19.06	28.87	35.26	40.40	6.12
	6.only pos.sim(F)	18.22	26.80	31.63	40.06	18.68	30.35	36.58	41.00	6.25
	7.only dep.sim(F)	19.35	26.81	32.90	42.77	18.45	29.20	35.11	40.91	<u>6.52</u>

"(I)" or "(F)" bebind the variants denote the augmentation modules which they belong to (i.e., instance-level or feature-level augmentation). The best scores are in orange and the second best ones are underlined

`M`, `L`). The reason is that the feature-level augmentation also needs to learn from the training data, and thus is inferior when lacking labeled instances. Therefore, in practice, we suggest a hierarchical augmentation framework by placing the feature-level augmentation on top of the instance-level augmentation.



(a) GloVe-ABSA. (b) BERT-ABSA. (c) GloVe-NER. (d) BERT-NER.

Figure 3 Impacts of the threshold τ in the instance-level augmentation

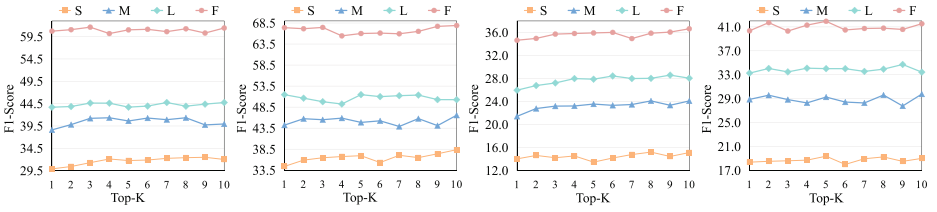
In variants 3∼4, we examine the effectiveness of different dependency paths for constructing the descriptions in the instance-level augmentation. In ABSA, the tail-related paths are more important than the head-related paths, but the opposite is true in NER. Since the tails of a given token could contain multiple tokens while its head is exactly a single token, tail-related paths show more diversity while head-related paths show higher accuracy. In ABSA, the polarity of an aspect term is determined by its contexts like verbs ("love") and adjectives ("good"), thus considering more related words is beneficial to the classification of sentiment. In contrast, the categories of named entities in NER only depend on themselves, and hence the accurate paths are more useful for generating synthetic instances.

In variants 5∼7, we examine the impacts of different similarities for retrieving the demonstrations in the feature-level augmentation. By only preserving one of three similarities, we can find that all the similarities are important and none of them can completely cover the others.

5.3 Parameter study

There are two key hyperparameters in D^{3}A: the threshold τ in constructing descriptions at the instance level and the number of demonstrations K at the feature level. Here we investigate their impacts by varying them in certain ranges and observing the performance trends.

Figure 3 shows the impacts of τ by varying its value in the range [0.1, 1.0] stepped by 0.1. Although the trends of curves are not very obvious, we can analyze by marking the best-performing τ in different sizes of training data. For example, with small (`S`) training data, diversity is more important since candidate mentions are rare, and the best results are achieved when $\tau \in [0.6, 1.0]$. On the contrary, with `full` (`F`) training data, $\tau \in [0.1, 0.4]$



(a) GloVe-ABSA. (b) BERT-ABSA. (c) GloVe-NER. (d) BERT-NER.

Figure 4 Impacts of the number of demonstrations K in the feature-level augmentation

Table 5 Comparison of gold instances and synthectic instances generated by MR and D^3A

Example
GOLD: But the **staff** was so horrible to us . (Restaurant)
MR: But the **tables** was so horrible to us .
D^3A: But the **maitre-d** was so horrible to us .
GOLD: Going to bring it to **service** today . (Laptop)
MR: Going to bring it to **17 ince screen** today .
D^3A: Going to bring it to **lg notebook service center** today .
GOLD: is up and ready for his last day on the punt until **Stakes Day** (WNUT16)
MR: is up and ready for his last day on the punt until **Army Run**
D^3A: is up and ready for his last day on the punt until **National POW/MIA Recognition Day**
GOLD: Win $ 100 **Visa** card ! [URL] (WNUT17)
MR: Win $ 100 **Ball Metal Container** card ! [URL]
D^3A: Win $ 100 **nintendo** card ! [URL]

brings promising performance since accuracy becomes dominant when candidate mentions are adequate.

Figure 4 shows the impacts of K by varying its value in the range [1, 10] stepped by 1. When more demonstrations are injected, the curves of GloVe-ABSA and GloVe-NER are generally upward. This trend is reasonable since GloVe embeddings contain limited knowledge and more demonstrations equal to more supporting information. While for BERT-ABSA and BERT-NER, only 1∼3 demonstrations can achieve satisfactory performance since the BERT backbone already embeds sufficient knowledge. In this case, the latter demonstrations with low similarity scores seem to be noisy and not informative.

5.4 A Closer look at D^3A

In this section, we take a closer look at D^3A. As shown in Table 5, we first present several synthetic instances generated by MR[7] and D^3A, and compare them with the gold instances to observe the synthetic quality in description-guided instance-level augmentation. Take the example from Restaurant as the example. For "staff" tagged as "`B-Positive`", MR replaces it with "tables" while D^3A replaces it with "maitre-d". Obviously, "maitre-d" is a more suitable mention here than "tables" to serve as a subject having the attitude "horrible". Thanks to the qualified synthetic instances, D^3A is powerful for the instance-level data augmentation as shown in the ablation study. We then examine the retrieved demonstrations in Table 6 to observe the influence of demonstration-guided feature-level augmentation. For a target token like "food", its encoded feature in the sequence tagger is augmented by related tokens like "meal, pizza, dessert". Therefore, it will be much easier than before for sequence taggers to recognize "food" as an aspect term.

[7]We choose MR as the representative method because it performs well in most cases, and also because MR adopts the mention replacement strategy which is of the same type as ours.

Table 6 Case study of tokens and their demonstrations in different datasets

Token	Demonstration
food (Restaurant)	meal 2.48 | pizza 2.43 | restaurant 2.42 | dessert 2.35 | fish 2.35 | soup 2.30 | chicken 2.29 | dinner 2.29 | seafood 2.29 | sushi 2.28
battery (Laptop)	keyboard 2.29 | power 2.27 | screen 2.20 | system 2.19 | life 2.12 | time 2.11 | charger 2.10 | warranty 2.10 | notebook 2.05 | hp 2.04
band (WNUT16)	fans 1.91 | games 1.90 | videos 1.88 | friends 1.84 | eagles 1.82 | ones 1.77 | books 1.74 | songs 1.73 | ears 1.72 | rappers 1.70
account (WNUT17)	status 2.26 | Website 2.25 | exchange 2.23 | message 2.19 | office 2.14 | risk 2.14 | system 2.13 | country 2.09 | track 2.09 | password 2.08

The value behind each demonstration denotes its similarity score with the target token

6 Conclusion

In this paper, we propose a description and demonstration guided data augmentation method D^3A for sequence tagging. By combining both instance-level and feature-level augmentation, D^3A can effectively improve the performance and generalization of sequence taggers. Specifically, at the instance level, we construct descriptions for mentions via head-related and tail-related dependency paths and generate reliable synthetic data. At the feature level, we retrieve demonstrations for tokens to enhance the learning capability of sequence taggers given limited training data. We conduct extensive experiments on NER and ABSA using different sizes of training sets. The results on both GloVe-based and BERT-based backbone sequence taggers demonstrate that D^3A can significantly improve the performance for sequence tagging tasks, especially in low-resource scenarios. In the future, we plan to investigate other methods for instance-level and feature-level augmentation, and generalize the data augmentation methods to more NLP tasks like relation extraction and event extraction.

Acknowledgements This work has been supported in part by the National Natural Science Foundation of China (NSFC) Projects (61572376, 62032016, 61972291).

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Zhuang Chen and Tieyun Qian. The first draft of the manuscript was written by Zhuang Chen and revised by Tieyun Qian. All authors read and approved the final manuscript.

Funding National Natural Science Foundation of China Projects 61572376, 62032016, 61972291.

Availability of data and material The data and material used in this paper have been uploaded at https://github.com/NLPWM-WHU/D3A.

Code availability The demo code of the proposed method in this paper has been uploaded at https://github.com/NLPWM-WHU/D3A.

Declarations

Financial interests The authors declare they have no financial interests.

Non-financial interests The authors declare they have no non-financial interests.

References

1. Asai, A., Hajishirzi, H.: Logic-guided data augmentation and regularization for consistent question answering. In: ACL, pp. 5642–5650 (2020)
2. Che, W., Zhao, Y., Guo, H., Su, Z., Liu, T.: Sentence compression for aspect-based sentiment analysis. IEEE ACM Trans. Audio Speech Lang. Process. **23**(12), 2111–2124 (2015)
3. Chen, Z., Qian, T.: Relation-aware collaborative learning for unified aspect-based sentiment analysis. In: ACL, pp. 3685–3694 (2020)
4. Dai, X., Adel, H.: An analysis of simple data augmentation for named entity recognition. In: COLING, pp. 3861–3867 (2020)
5. Derczynski, L., Nichols, E., van Erp, M., Limsopatham, N.: Results of the WNUT2017 shared task on novel and emerging entity recognition. In: NUT@EMNLP, pp. 140–147 (2017)
6. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL, pp. 4171–4186 (2019)
7. Ding, B., Liu, L., Bing, L., Kruengkrai, C., Nguyen, T.H., Joty, S.R., Si, L., Miao, C.: DAGA: data augmentation with a generation approach forlow-resource tagging tasks. In: EMNLP, pp. 6045–6057 (2020)
8. Fadaee, M., Bisazza, A., Monz, C.: Data augmentation for low-resource neural machine translation. In: ACL, pp. 567–573 (2017)
9. Feng, S.Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., Hovy, E.H.: A survey of data augmentation approaches for NLP. In: ACL Findings, vol. ACL/IJCNLP 2021, pp. 968–988 (2021)
10. Guo, D., Kim, Y., Rush, A.M.: Sequence-level mixed sample data augmentation. In: EMNLP (2020)
11. Huang, L., Sun, X., Li, S., Zhang, L., Wang, H.: Syntax-aware graph attention network for aspect-level sentiment classification. In: COLING, pp. 799–810 (2020)
12. Jiang, L., Yu, M., Zhou, M., Liu, X., Zhao, T.: Target-dependent twitter sentiment classification. In: ACL, pp. 151–160 (2011)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
14. Kobayashi, S.: Contextual augmentation: Data augmentation by words with paradigmatic relations. In: NAACL-HLT, pp. 452–457 (2018)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
16. Kruengkrai, C., Nguyen, T.H., Mahani, S.A., Bing, L.: Improving low-resource named entity recognition using joint sentence and token labeling. In: ACL, pp. 5898–5905 (2020)
17. Li, X., Bing, L., Li, P., Lam, W.: A unified model for opinion target extraction and target sentiment prediction. In: AAAI, pp. 6714–6721 (2019)
18. Lin, P., Yang, M., Lai, J.: Deep selective memory network with selective attention and inter-aspect modeling for aspect level sentiment classification. IEEE ACM Trans. Audio Speech Lang. Process. **29**, 1093–1106 (2021)
19. Lin, S., Gao, J., Zhang, S., He, X., Sheng, Y., Chen, J.: A continuous learning method for recognizing named entities by integrating domain contextual relevance measurement and Web farming mode of Web intelligence. World Wide Web **23**(3), 1769–1790 (2020)
20. Lin, Y., Fu, Y., Li, Y., Cai, G., Zhou, A.: Aspect-based sentiment analysis for online reviews with hybrid attention networks. World Wide Web **24**(4), 1215–1233 (2021)
21. Longpre, S., Lu, Y., Tu, Z., DuBois, C.: An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In: MRQA@EMNLP, pp. 220–227 (2019)
22. Luo, G., Huang, X., Lin, C.-Y., Nie, Z.: Joint entity recognition and disambiguation. In: EMNLP, pp. 879–888 (2015)
23. Luo, H., Li, T., Liu, B., Wang, B., Unger, H.: Improving aspect term extraction with bidirectional dependency tree representation. IEEE ACM Trans. Audio Speech Lang. Process. **27**(7), 1201–1212 (2019)
24. Ma, D., Li, S., Wu, F., Xie, X., Wang, H.: Exploring sequence-to-sequence learning in aspect term extraction. In: ACL, pp. 3538–3547 (2019)
25. Ma, D., Li, S., Zhang, X., Wang, H.: Interactive attention networks for aspect-level sentiment classification. In: IJCAI, pp. 4068–4074 (2017)
26. Manek, A.S., Shenoy, P.D., Mohan, M.C., Venugopal, K.R.: Aspect term extraction for sentiment analysis in large movie reviews using gini index feature selection method and SVM classifier. World Wide Web **20**(2), 135–154 (2017)
27. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL, pp. 55–60 (2014)

28. Mitchell, M., Aguilar, J., Wilson, T., Durme, B.V.: Open domain targeted sentiment. In: EMNLP, pp. 1643–1654 (2013)
29. Nie, Y., Tian, Y., Wan, X., Song, Y., Dai, B.: Named entity recognition for social media texts with semantic augmentation. In: EMNLP, pp. 1383–1391 (2020)
30. Passos, A., Kumar, V., McCallum, A.: Lexicon infused phrase embeddings for named entity resolution. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014, pp. 78–86 (2014)
31. Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., Clercq, O.D., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N.V., Kotelnikov, E.V., Bel, N., Zafra, S.M.J., Eryigit, G.: Semeval-2016 task 5: Aspect based sentiment analysis. In: NAACL-HLT, pp. 19–30 (2016)
32. Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., Androutsopoulos, I.: Semeval-2015 task 12: Aspect based sentiment analysis. In: SemEval, pp. 486–495 (2015)
33. Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., Manandhar, S.: Semeval-2014 task 4: Aspect based sentiment analysis. In: SemEval, pp. 27–35 (2014)
34. Popescu, A.-M., Etzioni, O.: Extracting product features and opinions from reviews. In: EMNLP, pp. 339–346 (2005)
35. Ratinov, L.-A., Roth, D.: Design challenges and misconceptions in named entity recognition. In: CoNLL, pp. 147–155 (2009)
36. Sahin, G.G., Steedman, M.: Data augmentation via dependency tree morphing for low-resource languages. In: EMNLP, pp. 5004–5009 (2018)
37. Sennrich, R., Haddow, B., Birch, A.: Improving neural machine translation models with monolingual data. In: ACL (2016)
38. Simard, P.Y., LeCun, Y., Denker, J.S., Victorri, B.: Transformation invariance in pattern recognition-tangent distance and tangent propagation. In: Neural Networks: Tricks of the Trade, pp. 239–27 (1996)
39. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
40. Strauss, B., Toma, B., Ritter, A., de Marneffe, M.-C., Xu, W.: Results of the WNUT16 named entity recognition shared task. In: NUT@COLING, pp. 138–144 (2016)
41. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR, pp. 1–9 (2015)
42. Vicente, I.S., Saralegi, X., Agerri, R.: Elixa: A modular and flexible ABSA platform. In: SemEval@NAACL-HLT, pp. 748–752 (2015)
43. Wang, K., Shen, W., Yang, Y., Quan, X., Wang, R.: Relational graph attention network for aspect-based sentiment analysis. In: ACL, pp. 3229–3238 (2020)
44. Wei, J.W., Zou, K.: EDA: easy data augmentation techniques for boosting performance on text classification tasks. In: EMNLP-IJCNLP, pp. 6381–6387 (2019)
45. Xu, H., Liu, B., Shu, L., Yu, P.S.: BERT post-training for review reading comprehension and aspect-based sentiment analysis. In: NAACL-HLT, pp. 2324–2335 (2019)
46. Xu, J., He, H., Sun, X., Ren, X., Li, S.: Cross-domain and semisupervised named entity recognition in chinese social media: A unified model. TASLP **26**(11), 2142–2152 (2018)
47. Xue, W., Li, T., Rishe, N.: Aspect identification and ratings inference for hotel reviews. World Wide Web **20**(1), 23–37 (2017)
48. Yan, H., Deng, B., Li, X., Qiu, X.: TENER: adapting transformer encoder for named entity recognition. CoRR arXiv:1911.04474 (2019)
49. Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
50. Zhang, M., Zhang, Y., Vo, D.-T.: Neural networks for open domain targeted sentiment. In: EMNLP, pp. 612–621 (2015)
51. Zhang, M., Qian, T.: Convolution over hierarchical syntactic and lexical graphs for aspect level sentiment analysis. In: EMNLP, pp. 3540–3549 (2020)
52. Zhang, R., Yu, Y., Zhang, C.: Seqmix: Augmenting active sequence labeling via sequence mixup. In: EMNLP, pp. 8566–8579 (2020)
53. Zhou, J.T., Zhang, H., Jin, D., Zhu, H., Fang, M., Goh, R.S.M., Kwok, K.: Dual adversarial neural transfer for low-resource named entity recognition. In: ACL, pp. 3461–3471 (2019)
54. Zhu, P., Chen, Z., Zheng, H., Qian, T.: Aspect aware learning for aspect category sentiment analysis. TKDD **13**(6) (2019)