Liu ZR, Liu Y. Exploiting unlabeled data for neural grammatical error detection. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(4): 758–767 July 2017. DOI 10.1007/s11390-017-1757-4

Exploiting Unlabeled Data for Neural Grammatical Error Detection

Zhuo-Ran Liu [1] and Yang Liu [2,3,4,5,*], *Member, CCF, ACM, IEEE*

[1] *School of Software, Beihang University, Beijing 100191, China*
[2] *State Key Laboratory of Intelligent Technology and Systems, Tsinghua University, Beijing 100084, China*
[3] *Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China*
[4] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*
[5] *Jiangsu Collaborative Innovation Center for Language Competence, Xuzhou 221009, China*

E-mail: liuzhuoran17@163.com; liuyang2011@tsinghua.edu.cn

Received December 20, 2016; revised May 19, 2017.

Abstract Identifying and correcting grammatical errors in the text written by non-native writers have received increasing attention in recent years. Although a number of annotated corpora have been established to facilitate data-driven grammatical error detection and correction approaches, they are still limited in terms of quantity and coverage because human annotation is labor-intensive, time-consuming, and expensive. In this work, we propose to utilize unlabeled data to train neural network based grammatical error detection models. The basic idea is to cast error detection as a binary classification problem and derive positive and negative training examples from unlabeled data. We introduce an attention-based neural network to capture long-distance dependencies that influence the word being detected. Experiments show that the proposed approach significantly outperforms SVM and convolutional networks with fixed-size context window.

Keywords unlabeled data, grammatical error detection, neural network

1 Introduction

Automatic grammatical error detection and correction for natural languages have attracted increasing attention, for a large number of non-native speakers are learning or using foreign languages. Take English as an example. There are a large number of English learners around the world who need instantaneous accurate feedback to help improve their writings[1]. In the domain of scientific paper writing in which English is the main language, authors also need effective grammar checkers to help them in composing scientific articles[2].

There have been several Shared Tasks addressing grammar errors in recent years. HOO-2011[3], HOO-2012[4], CoNLL-2013[5] and CoNLL-2014[6] Shared Task all aim to correct grammar errors. The AESW Shared Task[7] aims to identify sentence-level grammar errors. These Shared Tasks help advance the research of grammatical error detection and correction.

Despite these advances, the scarcity of annotated data is still a major limitation on the research of grammatical error detection and correction. Researchers need mass annotated data to train a grammar checker, but unfortunately for them, there are only a small amount of annotated corpora available in a limited number of domains. Most annotated corpora are in the domain of learner English, e.g., NUCLE[8] and CLC[9], and others are from domains such as scientific papers, e.g., AESW dataset[2]. In order to train their systems with enough data, researchers use multiple corpora instead of one corpus[10].

Data scarcity is partly due to difficulties in build-

Regular Paper

Special Issue on Deep Learning

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61522204 and 61331013 and the National High Technology Research and Development 863 Program of China under Grant No. 2015AA015407. This research is also supported by the National Research Foundation of Singapore under its International Research Centre@Singapore Funding Initiative and administered by the IDM (Interactive Digital Media) Programme.

*Corresponding Author

ing an elaborately annotated corpus needed for training of a grammatical error correction system, as described by the team that built NUS Corpus of Learner English[8]. In order to obtain a reliable annotation, the team set up a guideline for annotators so that corrections are consistent. To ensure that these annotations are available, several annotators proposed their correction independently, and annotations most agreed upon were selected. Such annotating process is labour-intensive and time consuming, and the quality of the corpus is subject to human judgment and other factors such as budget. For example, the team was unable to perform double annotation for the main corpus due to budget constraints. The team spent a long time (over half a year) to annotate only 1 414 essays.

Given these difficulties in building annotated corpus, we hope to utilize un-annotated error-free texts in unsupervised training of a grammatical error correction or grammatical error detection system. Previously, efforts have been made to explore how realistic grammatical errors could be counterfeited automatically from error-free texts and therefore obtain a large amount of annotated data[11-14]. We therefore follow the idea of building a corpus by generating artificial errors, since there are large numbers of un-annotated texts available and most of them are error-free. We explore two ways of artificial error generation, one of which is proved to be effective in our experiment.

Training a system to correct grammatical errors might be a more difficult task when there is no supervision, since there are numerous error types and our method to generate artificial errors might not be sophisticated enough to cover all of them. We thus focus on grammatical error detection instead of correction. It is natural to address this task as binary classification, in which we make prediction as to whether a word is grammatically correct.

2 Background

2.1 Problem Statement

The goal of word-level grammatical error detection is to identify grammar errors at the word level. For example, given a sentence shown below, a grammatical error detection system is expected to correctly identify the erroneous word "birds" highlighted by an underline:

An ugly <u>birds</u> was observed by the man yesterday.

The task of word-level grammatical error detection is formalized as such: given a sequence of token $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ as input, the error detector outputs its prediction $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_n)$ where $\boldsymbol{y}_i$ denotes the correctness of $\boldsymbol{x}_i$ in terms of grammaticality.

We address this problem as a binary classification problem. In order to predict $\boldsymbol{y}_t$ given the current word $\boldsymbol{x}_t$ and the whole sentence $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$, we need to find a function $g(\cdot)$ to calculate the conditional probability of each $\boldsymbol{y}_t$ given $\boldsymbol{x}_t$ and the whole input sequence $\boldsymbol{X}$:

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t) = g(\boldsymbol{x}_t, \boldsymbol{X}),$$

where

$$\boldsymbol{y}_t = \begin{cases} 1, & \text{if } \boldsymbol{x}_t \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

Our aim is to build a suitable classification model for $g(\cdot)$.

2.2 SVM Model for Error Detection

A natural approach is to use support vector machine (SVM) to perform classification[15-16]. SVM is trained given a training dataset in the form of $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i$ represents a token with a set of selected linguistic features, and $\boldsymbol{y}_i$ denotes the grammatical correctness of the token. It finds a maximum-margin hyperplane that separates correct words from incorrect ones.

The problem with this approach is that we need to manually design features in $\boldsymbol{x}_i$. Since human are unable to tell precisely which features are relevant, human-designed features are inadequate in some aspects while being redundant in others. As a result, these designed features are unable to capture all regularities, which might hurt the performance of our error detector.

2.3 Convolution Network with Fixed Window Size

To circumvent the problem with feature engineering, a natural thought is to utilize the capability of neural networks in automatic feature extraction[17]. The simplest way is to take into consideration a fixed-size window of words around the current word as its context by applying temporal convolution over the fixed-size window. In the example sentence given in Subsection 2.1, when considering the grammatical correctness of the word "was" given a context window of size 3, the context window would be "birds was observed". The assumption that underlies this method is that only neighbouring words are grammatically related to the current word.

Here we formalize the method of neural network with a fixed-size window. Given a word $\boldsymbol{x}_i$, its context is:

$$\boldsymbol{c}_i = (\boldsymbol{x}_{i-w/2}, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_{i+w/2}).$$

Let $f(\cdot)$ denote a temporal convolution operation with the input frame size equal to the dimension of $\boldsymbol{x}_i$, the output frame size equal to 1, and the kernel width equal to the size of a fixed-size window. A score $\boldsymbol{s}_i$ of the current word $\boldsymbol{x}_i$ is calculated by $\boldsymbol{s}_i = f(\boldsymbol{c}_i)$ which represents grammatical features within the window. This score then goes through a sigmoid layer and yields the probability of $\boldsymbol{y}_i$: $p(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{c}_i) = \sigma(\boldsymbol{s}_i)$.

The first problem with this method is that it is incapable of capturing long-distance dependency. With a fixed window size, the error detector is unable to take into consideration word contexts beyond the window size, while long-distance grammatical dependency is quite common a phenomenon. For example, in order to determine whether "was" is incorrect, we would need to take "yesterday" into consideration, which requires a large size of context window.

Another problem is that all words within the context window are taken into consideration indiscriminately. In the example above, "was" might not care about what was done to the birds when determining the verb tense, but "observed" is given equal attention regardless of the fact that it has no influence on verb tense.

3 Approach

3.1 Model Architecture

Our intuition is to first encode the input sequence into a sequence of hidden states which contain relevant grammatical information, and then make predictions given a word and its context (see Fig.1). Thus our model consists of two parts: an encoder that adopts a typical architecture of bi-directional LSTM network[18], and a classifier that makes predictions based on hidden states of the encoder.

3.1.1 *Encoder*

The encoder takes as input a sentence S of length n, represented by a sequence of vector $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$. In an LSTM recurrent neural network, input $\boldsymbol{X}$ is processed through time and produces a series of memory states $(\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_n)$ and hidden states $(\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n)$. In order to counterbalance the impact of time on hidden states we process the input $\boldsymbol{X}$ twice, forward and backward, to fully encode the information that the classifier needs.

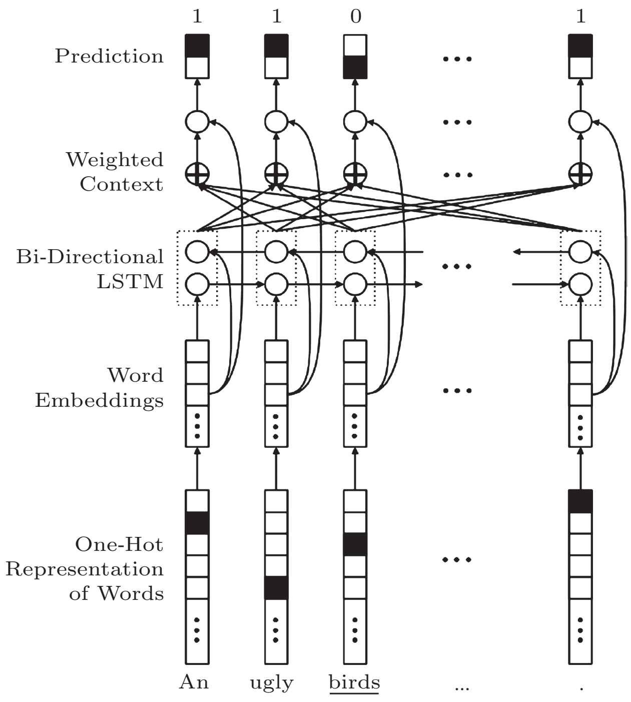

Fig.1. Model architecture. The input of the network is a sentence "an ugly birds was observed by the man yesterday" in the form of one-hot representation. The representation is then converted into continuous word-embeddings and encoded by a bi-directional LSTM encoder. The encoded information is reweighted by an intra-attention mechanism at each time-step, on which the classifier judges the grammaticality of each word.

The forward LSTM updates its memory state $\overrightarrow{\boldsymbol{c}_i}$ and hidden state $\overrightarrow{\boldsymbol{h}_i}$ at each time-step t:

$$[\overrightarrow{\boldsymbol{h}_t}; \overrightarrow{\boldsymbol{c}_t}] = \overrightarrow{LSTM}([\overrightarrow{\boldsymbol{h}_{t-1}}; \overrightarrow{\boldsymbol{c}_{t-1}}]).$$

Similarly, memory state $\overleftarrow{\boldsymbol{c}_i}$ and hidden state $\overleftarrow{\boldsymbol{h}_i}$ are updated by the backward LSTM at time-step t:

$$[\overleftarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{c}_t}] = \overleftarrow{LSTM}([\overleftarrow{\boldsymbol{h}_{t+1}}; \overleftarrow{\boldsymbol{c}_{t+1}}]).$$

The encoder outputs a hidden tape $\widetilde{\boldsymbol{h}} = (\widetilde{\boldsymbol{h}_1}, \widetilde{\boldsymbol{h}_2}, ..., \widetilde{\boldsymbol{h}_n})$, where

$$\widetilde{\boldsymbol{h}_t} = \begin{bmatrix} \overrightarrow{\boldsymbol{h}_t} \\ \overleftarrow{\boldsymbol{h}_t} \end{bmatrix},$$

with $[\cdot]$ denoting the concatenation of vectors.

3.1.2 *Classifier with Intra-Attention*

To predict whether the word at time-step t is grammatically problematic, the classifier computes a score given the current word $\boldsymbol{x}_t$ and its context $\boldsymbol{a}_t$. This score $\boldsymbol{s}_t$ then goes through a sigmoid layer and makes a binary prediction, with 1 denoting grammatically correct and 0 denoting incorrect. Note that the classifier does not hold its own state as a decoder does in a traditional encoder-decoder architecture.

To address the problem of long-distance dependency, we incorporate an intra-sentence attention mechanism[19] in our classifier, where all hidden states of the encoder are taken into consideration and the attention of the classifier on all positions of the sentence is dynamically adapted. To describe formally, we compute the context $\boldsymbol{a}_t$ around the word $\boldsymbol{x}_t$ as an attention-weighted sum of $\{\widetilde{\boldsymbol{h}_1}, \widetilde{\boldsymbol{h}_2}, ..., \widetilde{\boldsymbol{h}_n}\}$:

$$\boldsymbol{a}_t = \sum_i \alpha_{t,i} \cdot \widetilde{\boldsymbol{h}_i},$$

where

$$\alpha_{t,i} = \frac{\exp(\boldsymbol{E}_{t,i})}{\sum_j \exp(\boldsymbol{E}_{t,j})},$$
$$\boldsymbol{E}_{t,i} = \widetilde{\boldsymbol{h}_t} \cdot \widetilde{\boldsymbol{h}_i}.$$

Vector $\boldsymbol{a}_t$ represents the grammatical and semantic context at position t. A word is considered to be grammatically erroneous if the word $\boldsymbol{x}_t$ does not fit into the current context, i.e., it is incompatible to place $\boldsymbol{x}_t$ at position t given the context $\boldsymbol{a}_t$. The score $\boldsymbol{s}_t$ is computed as follows:

$$\boldsymbol{s}_t = \boldsymbol{x}_t^{\mathrm{T}} \cdot \boldsymbol{W} \cdot \boldsymbol{a}_t + \boldsymbol{b},$$

where $\boldsymbol{b}$ is the bias.

Then the probability can be calculated as

$$p(\boldsymbol{y}_t | \boldsymbol{x}_t, \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}) = \sigma(\boldsymbol{s}_t).$$

By incorporating intra-attention mechanism, we provide a latent structure for the model to learn grammatical relations between words. This makes a lot of sense because the grammaticality of a word is dependent more on the words that have strong grammatical relation with it, while other words are negligible when making predictions. For example in Fig.1, when the model tries to determine whether "birds" is correct in terms of the number of the noun, it will pay a strong attention to "An", which indicates that the noun "bird" should take its singular rather than plural form.

3.2 Noise Generation

Traditionally, a large set of $\{(\boldsymbol{X}^{(n)}, \boldsymbol{Y}^{(n)})\}_{n=1}^N$ is needed to effectively train such a grammatical error detector. However when only $\{\boldsymbol{X}^{(n)}\}_{n=1}^N$ is given, the key issue now becomes how to obtain the corresponding $\{\boldsymbol{Y}^{(n)}\}_{n=1}^N$.

We adopt the idea of using artificial errors for training. It is crucial to find a suitable algorithm for the error generator to produce realistic grammatical errors, since the performance of the model relies heavily on the paradigm it observed during training. Since our task is to detect grammatical errors on the word level, we only consider substitution errors. We compare two ways of substituting the original word for an erroneous one.

3.2.1 Uniform Random Substitution

The simplest way is to substitute a word in a random position with a random word from the vocabulary. The problem with this approach is that some artificial errors generated in this way are apparently irrelevant.

For example, it could substitute a word from the sentence "An ugly bird was observed by the man yesterday." to generate such a sentence as "An ugly bird was **dog** by the man yesterday." One potential problem is that it might be too easy for our classifier to discriminate such erroneous words from the correct ones.

3.2.2 Substitution with Linguistic Knowledge

We carefully examined a number of erroneous paradigms and found some characteristics common to all grammatical errors, regardless of the terminology and commonly seen patterns of the domain. To briefly summarize it, errors usually appear when a correct word is substituted by another word, which comes from a finite set of words linguistically related to it, because this set of words possess the same lemma or the same part-of-speech tag.

There is an inexhaustible list of how linguistic knowledge works in substitution. Here we only present several examples in Table 1.

Combining these two methods, uniform random substitution and substitution with lingustic knowledge, we are able to generate 16 types of grammatical errors out of 28 specified by CoNLL-2014 Shared Task[6]. Error types that can be generated are: Vt, Vm, Vform, SVA, ArtOrDet, Nn, Npos, Pform, Pref, Prep, Wci, Wform, Spar, Trans, Mec, Others. Most of remaining error types we are unable to generate are semantic errors (Smod, Rloc-, UM), or style problems (Wa, Wtone, Cit), or sentence level problems (Srun, Sfrag, WOinc, WOadv).

Details of artificial error generation process that incorporates linguistic knowledge are described in Algorithm 1, which formalizes the construction of substitution set, and in Algorithm 2, which formalizes the

Table 1. Examples of Substitution with Linguistic Knowledge

POS Tag	Original	Noise	Example
VB	built	build, builds, building, ...	Workers built the park centuries ago. Workers **build** the park centuries ago.
NN	eggs	egg	All eggs were put into the same basket. All **egg** were put into the same basket.
DT	an	a, this, these, ...	There is an apple on the table. There is **a** apple on the table.
RB	suitably	suitable	Candidates must be suitably qualified students. Candidates must be **suitable** qualified students.
IN	of	in, by, for, at, ...	This book consists of 12 chapters. This book consists **by** 12 chapters.

process of error generation by using the substitution set built in Algorithm 1.

Algorithm 1. Building Substitution Set

```
1: PoS-tag the input text
2: Build a dictionary $\mathbb{D}$ of $(token, PoS - tag)$
3: for all $(token, pos)$ in $\mathbb{D}$ do
4:     if pos in {CC} or {DT, PDT} or {PRP, PRP$} or {IN, TO, RP} or {WDT, WP, WP$, WRB} then
5:         Add token to the corresponding substitution set $\mathbb{C}_i$
6:     else if pos in {NN, NNP, NNPS, NNS} or {VB, VBD, VBG, VBN, VBP, VBZ} then
7:         lemma ← Lemmatise token
8:         Add token to the corresponding substitution set $\mathbb{C}_i$
9:     else if pos in {JJ, JJR, JJS} or {RB, RBR, RBS} then
10:        stem ← Stem token
11:        Add token to the corresponding substitution set $\mathbb{C}_i$
12:    end if
13: end for
```

Algorithm 2. Error Generation

```
1: for all sentences $S$ in training text do
2:     Get word $w$ at a random position of $S$
3:     $w' \leftarrow w$
4:     Search for substitution set $\mathbb{S}_i$ that contains $w$
5:     if such $\mathbb{C}_i$ does not exist or $\mathbb{C}_i$ contains only 1 element then
6:         while $w' == w$ do
7:             $w' \leftarrow$ Select a random word from dictionary $\mathbb{D}$
8:         end while
9:     else
10:        while $w' == w$ do
11:            $w' \leftarrow$ Select a random word from $\mathbb{C}_i$
12:        end while
13:    end if
14:    Replace $w$ in $S$ with $w'$
15: end for
```

4 Experiments

4.1 Settings

4.1.1 Data

We use data mainly from three sources (Table 2):

- ACL Anthology[1] (ACL): training set;
- AESW Shared Task Dataset (AESW)[7]: development and test sets;
- CCL Anthology[2] (CCL): development and test sets.

Table 2. Statistics of Datasets Used in the Experiments

	Set	Tok.	Pct. (%)	Vocab.	Sent.
AESW	Development	24.4k	6.0	4.1k	1.0k
	Test	24.7k	5.9	4.1k	1.0k
CCL	Development	2.6k	5.5	892	125
	Test	2.8k	5.2	934	126
ACL	Train	60.4M	4.9	166.5k	2.9M

Note: Tok. stands for the number of tokens, Pct. stands for percentage of tokens that are marked incorrect, Vocab. stands for the size of vocabulary, and Sent. stands for total number of sentences.

For training set, we use sentences from papers that appear in ACL Anthology. We crawl all papers up to year 2015, and then select sentences that end with a period, with a length of longer than 5 but no longer than 50, which may contain several clauses separated by commas, colons or semicolons. Formulae and references are excluded, numbers are substituted with a special $\langle num \rangle$ token, and parentheses are removed together with the contents in between. We limite the vocabulary to tokens with at least a word-frequency of 2 to eliminate most spelling errors, and replace all OOVs with a special $\langle unk \rangle$ token.

To corroborate that the model trained by us actually works with realistic grammatical errors, we use two human-annotated datasets as our development and test set.

The first one is the test set of AESW 2016 Shared Task, but we only use a portion of the erroneous sentences from paragraphs with the attribute of "domain=Computer Science"; we converte the data format by preserving all words between "$\langle del \rangle \langle /del \rangle$" and marking them as incorrect, while removing those between "$\langle ins \rangle \langle /ins \rangle$".

[1]http://www.aclweb.org/anthology/, May 2017.

[2]http://www.cips-cl.org/anthology, May 2017.

For example, if the original annotated sentence is

"More discussions $\langle del\rangle$about$\langle /del\rangle$ $\langle ins\rangle$on$\langle /ins\rangle$ these issues will be provided in the remainder of the monograph.",

we convert it into the form of:

"More discussions **about** these issues will be provided in the remainder of the monograph."

The second human-annotated dataset is some erroneous sentences from papers in CCL Anthology annotated by us. These papers contain grammatical errors since most of them are written in Chinese.

A potential problem with the way we annotate our dataset is that the case of multiple gold-standard is not well addressed. For example, in the sentence "An ugly birds was observed by the man yesterday.", reporting errors on any one or several of "an", "birds", or "was" should be a correct prediction. However as we are only annotating one word as erroneous, other correct predictions are counted as incorrect, which affects model performance.

4.1.2 Baselines

To the best of our knowledge, word-level grammatical error detection task has never been researched before. Thus we use the two methods described in Subsection 2.2 and Subsection 2.3 respectively. We compare our method with two baselines, both of which are trained on the ACL training set with artificial errors. We further compare our method with yet another baseline — RNNLM[20], which is trained on un-annotated error-free ACL sentences. Ultimately, we build up three baselines and compare our method with them:

- support vector machine (SVM)[16],
- convolutional network (Conv)[17],
- recurrent neural network based language model (RNNLM)[20].

In our baseline SVM, we take into consideration the context in a fixed-size window of size 5 around the current word. The SVM classifier then gives the prediction of whether the current word is grammatically correct in the sentence. We first train an n-gram model with KenLM[21] on the whole training set without artificial errors, with n up to 3. We then use n-gram scores as the input features into the SVM. In our experiment we use the open-source tool LibLinear[22].

As described in Subsection 2.3, we build the baseline referred to as Conv in the following way. We use word-embeddings pre-trained using Word2Vec model in gensim[23], the dimensionality of which is set to 50 empirically. A temporal convolution is performed over a window of fixed size 3. The kernel width is set to the size of a fixed-size window. This model is implemented using Torch7③.

An alternative way to make use of unlabeled error-free data is by training a language model on them. We adopt the state-of-the-art language model — RNNLM[20] for one of our baselines. An RNN-based language model is first trained on error-free texts with RNNLM-Toolkit[24], which models the probability of a word given a preceding sequence: $p(w_t|w_1w_2\cdots w_{t-1})$. An intuition is that when we substitute w_t with a random word, the probability would decrease if w_t is grammatical. Following this criterion we build a classifier as our baseline.

4.1.3 BiLSTM with Intra-Attention

Our model described in Subsection 3.1 is implemented using Tensorflow④. We use cross-entropy as our loss function to optimize. We perform gradient clipping by global norm[25] with the function provided in Tensorflow. The dimension of word-embedding and hidden states is set to 150, as a trade-off between performance and training time. The word-embedding matrix is initialized with random uniform distribution within the range of ± 0.05.

4.2 Results and Discussion

Table 3 and Table 4 present the results: precision (P) and recall (R), of the experiments of three baselines (SVM, Conv, and RNNLM) and our model (BiLSTM), using uniform random errors (uni.) or errors counterfeited with linguistic knowledge (ling.).

Table 3. Performance on the AESW Test Set Measured by $F_{0.5}$(%)

Method	Noise	P	R	$F_{0.5}$
SVM	uni.	13.53	6.27	10.99
	ling.	12.51	7.15	10.88
Conv	uni.	6.25	50.10	7.57
	ling.	18.13	4.46	11.24
RNNLM	-	8.95	21.52	10.13
BiLSTM	uni.	17.16	5.39	11.95
	ling.	18.71	7.48	**14.40**

③http://torch.ch/, May 2017.

④https://www.tensorflow.org/, May 2017.

Table 4. Performance on the CCL Test Set Measured by $F_{0.5}$(%)

Method	Noise	P	R	$F_{0.5}$
SVM	uni.	7.40	1.34	3.89
	ling.	6.25	1.34	3.61
Conv	uni.	5.66	57.43	6.91
	ling.	6.66	0.67	2.40
RNNLM	-	11.43	23.48	12.74
BiLSTM	uni.	16.00	2.68	8.03
	ling.	21.05	8.05	**15.91**

From the two tables we can see that our model outperforms the two baselines on both human-annotated datasets (AESW and CCL). The $F_{0.5}$ scores might seem low, but they are actually good results since these models are trained without supervision.

4.2.1 *Effect of Error Types*

If we focus on the task of detecting a limited number of error types (verb form, noun number, preposition misuse, article misuse), the model's performance is better on the CCL test set, but is weaker or generally unchanged on the AESW test set. This is probably because in the annotating phase of the CCL test set, we focuse heavily on these common types of errors and some other types of errors are neglected. The results are shown in Table 5 and Table 6, where w/o means without, and w/ means with.

Table 5. Comparison on the AESW Test Set

Error Type	Attention	P	R	$F_{0.5}$
All	w/o	14.84	6.61	11.88
	w/	18.71	7.48	14.40
Limited	w/o	19.87	4.25	11.45
	w/	18.48	6.27	13.31

Table 6. Comparison on the CCL Test Set

Error Type	Attention	P	R	$F_{0.5}$
All	w/o	23.40	7.38	16.32
	w/	21.05	8.05	15.91
Limited	w/o	26.00	8.72	18.62
	w/	27.90	16.10	24.34

4.2.2 *Effect of Attention*

To verify our intra-attention help improve model performance, we removed the attention and performed the same experiment. Comparison of models with and without attention is shown in Table 5 and Table 6.

Though the intra-attention mechanism works in improving overall performance, in some individual cases it may fail.

We believe that a more sophisticated way of error generation is needed, because currently only those positions where substitution happens have a chance to be labeled incorrect ("on-site-error" paradigms). But for the model to learn grammatical relations by attention mechanism, we need massive paradigms where substitution causes another position to be labeled incorrect ("off-site-error" paradigms).

Take this sentence as an example:

"An ugly bird was observed by the man yesterday."

If we substitute "ugly" with "beautiful", our system will automatically annotate "beautiful" as incorrect (on-site-error):

"An **beautiful** bird was observed by the man yesterday." But our model will never know why it is incorrect. What we need instead is off-site-errors:

"An **beautiful** bird was observed by the man yesterday."

So that the model knows "beautiful" is compatible with "a" but not with "an".

Unfortunately our method does not provide such a mechanism to massively produce "off-site-error" paradigms; therefore our model has to rely on very few coincidentally generated "off-site-error" paradigms which are too sparse.

Our current method also introduces some substitutions which should not be counted as errors. For example, if "yesterday" is substituted by "today":

"An ugly bird was observed by the man **today**."

"today" is annotated as incorrect under our method, while it does not actually constitute grammatical errors, which therefore hurts model performance to some degree.

4.2.3 *Similarity Between Artificial and Real Grammatical Errors*

Artificial grammatical errors are generated for training, while human-annotated texts are used for test sets. In order to measure the similarity between artificial training data and real ungrammatical sentences, we provide statistics on the syntactic information of training set and test sets.

As shown in Table 7, the distribution of ungrammatical words in terms of part-of-speeches is generally consistent between artificial data (ACL training set) and real data (AESW and CCL test set).

Table 7. Percentage of Words with Different Part-of-Speeches That Cause Grammatical Errors (in Descending Order)

Real		Artificial
AESW	CCL	ACL
NOUN(28.47)	VERB(33.55)	NOUN(45.65)
VERB(11.47)	NOUN(20.80)	VERB(22.44)
ADJ(9.24)	PREP(16.10)	ADJ(22.22)
PREP(8.77)	DET(6.04)	ADV(4.47)
DET(7.01)	ADJ(5.36)	PREP(3.50)
ADV(4.18)	WH(3.35)	DET(0.64)
PRON(1.28)	ADV(2.68)	PRON(0.23)
WH(1.07)	PRON(1.34)	WH(0.18)
CONJ(0.87)	CONJ(0.67)	CONJ(0.16)

Note: Real PoS-tag is not shown in this table but merged into several classes: e.g. "VERB(33.55)" means words with PoS-tags "VB", "VBD", "VBG", "VBN", "VBP" and "VBZ" count for 33.55% of all erroneous words.

4.2.4 *Using Human-Annotated Data in Training*

To verify our method of error generation is useful for training an error detection model, we train and test a model with the same architecture on a blender of artificial data and the original training data of AESW Shared Task. To maintain the consistency of the text domain, we extract only sentences in "Computer Science" domain with errors (approx. 30k sentences) and add them into the training set to form a blender of artificial and real data. Experiments show that the performance of the model trained on only artificial data is comparable to that trained on the blender (Table 8).

Table 8. Performance of the Model Trained on a Blender of Artificial and Real Data

Testset	P	R	$F_{0.5}$
AESW	13.84	3.91	9.18
CCL	24.00	8.05	17.19

4.2.5 *Testing Model in a Realistic Scenario*

In real-life setting, the proportion of sentences that contain grammatical errors depends on the proficiency in the language of the persons who produce those sentences. We blend grammatical sentences with ungrammatical ones with different ratios to form several test sets and evaluated our model on them.

Results (Table 9) show that model performance deteriorates with the increase of the percentage of the grammatical sentences in the test set, which is possibly because an artificial error is introduced in every sentence of the training dataset. Error rate of training data will have to be considered in future research.

Table 9. Performance on the Test Sets with Different Error Rates

Error (%)	Test Set	P	R	$F_{0.5}$
100	AESW	18.71	7.48	14.40
	CCL	21.05	8.05	15.91
50	AESW	9.72	7.48	9.17
	CCL	11.53	8.05	10.61
10	AESW	2.04	7.48	2.39
	CCL	2.42	8.05	2.82

Note: Error (%) denotes the percentage of sentences that contain at least one grammatical error.

4.2.6 *Examples*

Our model is found to perform well in some cases, while failing to identify others. To analyse what type of errors it deals with well, and the reasons that cause its failure, we sample some predictions as presented below in Table 10. Note that the table contains only a partial list of error types our model detected. Since we only detect errors without inferring their types, we are unable to provide the full list of error types our model is able to detect.

Table 10. Examples of Model Predictions

Error Type	Example
Collocation	(Ex.1) In **additions** , we present an in-depth analysis that provides valuable insight into the characteristics of alternative solutions.
Morphology	(Ex.2) In our work, lexical level features include the two entities, their NER tags, and the **neighbor** tokens of these two entities.
Genre	(Ex.3) For the purpose of this study, we focus on one of our live **broadcast**, Premier Wen Talks Online with Citizens on **Feb** 28, 2009.
Domain	(Ex.4) Note that, the communication cost of the PAROS layer is constant and dependent on the network size and the gossiping period.
Wrong position	(Ex.5) In summary, filtering the sentences whose polarities **opposite** to the overall orientation is significant for constructing a high quality training set.
Other	(Ex.6) Compared with other methods, the **our** heterogeneous graph method improves the results significantly.

Note: Incorrect words are in bold face, and errors detected by our model are highlighted by underlines.

It works well with collocations, as in Ex.1. It also works well with morphological problems, as Ex.2 shows. However it is incapable of detecting errors of genre as in Ex.3. In Ex.4, it mistook as incorrect the words out of the domain of the training data. It is apparent in Ex.5 that our model is aware of the missing "are" between "polarities" and "opposite", but it reports errors at a different position. There are other error types our model did not handle well with, such as redundant determiners as in Ex.6.

5 Related Work

5.1 Grammatical Error Detection and Correction

Several Shared Tasks on grammatical error detection or correction have been carried out in recent years, including HOO-2011[3], HOO-2012[4], CoNLL-2013[5], and CoNLL-2014[6]. These four Shared Tasks all focus on grammatical error correction of English written by non-native speakers. The AESW Shared Task[7] proposes to evaluate scientific writing automatically based on sentence-level error identification. Different from these Shared Tasks, we focus on word-level grammatical error detection, which is a pilot step towards unsupervised approach to error correction.

To address the issue of grammatical errors, researchers explored and utilized various methods. For example classification method is used by the top ranking team[26] in CoNLL-2013 Shared Task. The top ranking team[10] of CoNLL-2014 Shared Task incorporated in the team's system a statistical machine translation (SMT) component, which translates erroneous English into correct English. With the development of neural machine translation (NMT) and attention mechanism[19], the top team[27] of AESW Shared Task adopted the NMT approach to grammatical error correction.

In comparison, we adopt a typical architecture of bi-directional LSTM[18] on the encoder side[28], but replace the decoder with a classifier. Since error types are not given in unsupervised training, our classifier does not infer error types but only makes binary predictions.

5.2 Error Generation

To obtain enough training data, various approaches have been employed to generate artificial errors. Markov logic network is used for statistical grammar error simulation[13]. An automatic tool for error generation was developed[12], which takes as input a corpus and error generation rules. Error inflation is used in UI system[29] in HOO-2012 Shared Task, and a similar method was performed on Japanese[14]. To enlarge the size of training set, artificial errors were injected into the corpus by Yuan and Felice in CoNLL-2013 Shared Task[30]. Later Felice and Yuan further researched the probabilistic manner of artificial error generation with linguistic information[11].

Different from [12] which requires a set of rules to work, we build substitution set automatically from un-annotated corpus based on POS tag or lemma. To compare with [11, 29-30] whose methods of error generation are based on annotated corpus, we use only un-annotated error-free texts without supervision.

5.3 RNNs and LSTM Units

Recurrent neural network (RNN) with long short-term memory (LSTM) or gated recurrent unit (GRU) has shown a mighty capability to encode information over long sequences[28]. The attention mechanism has enabled a bi-directional RNN with GRU to achieve even better performance in machine translation[19] by allowing the decoder to explicitly make use of the memory of the encoder. Upon the emergence of attention mechanism, it has been applied to many NLP topics other than machine translation. Grammatical error correction is of no exception. Schmaltz *et al.* used a uni-directional LSTM network with attention mechanism and achieved the best performance in the AESW Shared Task[27].

Different from [27], we do not generate a target sentence since we do not attempt to correct errors. Therefore we replace the decoder with a binary classifier, which takes into consideration the information from the BiLSTM encoder.

6 Conclusions

In our work, we explored unsupervised word-level grammatical error detection using only un-annotated corpus as training data. We showed that it is a viable way for machines to learn grammatical relations and to predict grammatical errors. This inspires us to further extend the unsupervised approach to grammatical error correction. In the future, we plan to investigate novel methods for generating artificial errors to enable our model to learn better intra-sentence attention.

References

[1] Dahlmeier D, Ng H T. A beam-search decoder for grammatical error correction. In *Proc. the Joint Conf. Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, July 2012, pp.568-578.

[2] Daudaravičius V. Automated evaluation of scientific writing: AESW Shared Task proposal. In *Proc. the 10th Workshop on Innovative Use of NLP for Building Educational Applications*, June 2015, pp.56-63.

[3] Dale R, Kilgarriff A. Helping our own: The HOO 2011 pilot Shared Task. In *Proc. the 13th European Workshop on Natural Language Generation*, September 2011, pp.242-249.

[4] Dale R, Anisimo I, Narroway G. Hoo 2012: A report on the preposition and determiner error correction Shared Task. In *Proc. the 7th Workshop on Building Educational Applications Using NLP*, June 2012, pp.54-62.

[5] Ng H T, Wu S M, Wu Y B, Hadiwinoto C, Tetreault J R. The CoNLL-2013 Shared Task on grammatical error correction. In *Proc. the 17th Conf. Computational Natural Language Learning*: *Shared Task*, August 2013.

[6] Ng H T, Wu S M, Briscoe T, Hadiwinoto C, Susanto R H, Bryant C. The CoNLL-2014 Shared Task on grammatical error correction. In *Proc. the 18th Conf. Computational Natural Language Learning*: *Shared Task*, July 2014.

[7] Daudaravicius V, Banchs R E, Volodina E, Napoles C. A report on the automatic evaluation of scientific writing Shared Task. In *Proc. the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, June 2016, pp.53-62.

[8] Dahlmeier D, Ng H T, Wu S M. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proc. the 8th Workshop on Innovative Use of NLP for Building Educational Applications*, June 2013, pp.22-31.

[9] Nicholls D. The Cambridge learner corpus-error coding and analysis for lexicography and ELT. In *Proc. Corpus Linguistics Conf.*, March 2003, pp.572-581.

[10] Felice M, Yuan Z, Andersen Ø E, Yannakoudakis H, Kochmar E. Grammatical error correction using hybrid systems and type filtering. In *Proc. the 18th Conf. Computational Natural Language Learning*: *Shared Task*, July 2014, pp.15-24.

[11] Felice M, Yuan Z. Generating artificial errors for grammatical error correction. In *Proc. Student Research Workshop at the 14th Conf. the European Chapter of the Association for Computational Linguistics*, April 2014, pp.116-126.

[12] Foster J, Andersen Ø E. GenERRate: Generating errors for use in grammatical error detection. In *Proc. the 4th Workshop on Innovative Use of NLP for Building Educational Applications*, June 2009, pp.82-90.

[13] Lee S, Lee G G. Realistic grammar error simulation using Markov logic. In *Proc. ACL-IJCNLP Conference Short Papers*, August 2009, pp.81-84.

[14] Imamura K, Saito K, Sadamitsu K, Nishikawa H. Grammar error correction using pseudo-error sentences and domain adaptation. In *Proc. the 50th Annual Meeting of the Association for Computational Linguistics*: *Short Papers*, July 2012, pp.388-392.

[15] Boser B E, Guyon I M, Vapnik V N. A training algorithm for optimal margin classifiers. In *Proc. the 5th Annual Workshop on Computational Learning Theory*, July 1992, pp.144-152.

[16] Cortes C, Vapnik V. Support-vector networks. *Machine Learning*, 1995, 20(3): 273-297.

[17] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. the 25th Int. Conf. Machine Learning*, July 2008, pp.160-167.

[18] Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation*, 1997, 9(8): 1735-1780.

[19] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. *arXiv: 1409.0473*, 2014. https://arxiv.org/abs/1409.047, May 2017.

[20] Mikolov T, Karafiát M, Burget L, Černocký J, Khudanpur S. Recurrent neural network based language model. In *Proc. the 11th Annual Conf. the Int. Speech Communication Association*, September 2010, pp.1045-1048.

[21] Heafield K. KenLM: Faster and smaller language model queries. In *Proc. the 6th Workshop on Statistical Machine Translation*, July 2011, pp.187-197.

[22] Fan R E, Chang K W, Hsieh C J, Wang X R, Lin C J. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 2008, 9: 1871-1874.

[23] Řehůrek R, Sojka P. Software framework for topic modelling with large corpora. In *Proc. the LREC Workshop on New Challenges for NLP Frameworks*, May 2010, pp.46-50.

[24] Mikolov T, Kombrink S, Deoras A, Burget L, Honza J, Černocký J. RNNLM-recurrent neural network language modeling toolkit. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, December 2011.

[25] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In *Proc. the 30th Int. Conf. Machine Learning*, June 2013, pp.III-1310-III-1318.

[26] Rozovskaya A, Chang K W, Sammons M, Roth D. The university of Illinois system in the CoNLL-2013 Shared Task. In *Proc. the 17th Conf. Computational Natural Language Learning*: *Shared Task*, August 2013, pp.13-19.

[27] Schmaltz A, Kim Y, Rush A M, Shieber S M. Sentence-level grammatical error identification as sequence-to-sequence correction. *arXiv: 1604.04677*, 2016. https://arxiv.org/abs/1604.04677, May 2017.

[28] Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. the Conf. Empirical Methods in Natural Language Processing* (*EMNLP*), October 2014, pp.1724-1734.

[29] Rozovskaya A, Sammons M, Roth D. The UI system in the HOO 2012 Shared Task on error correction. In *Proc. the 7th Workshop on Building Educational Applications Using NLP*, June 2012, pp.272-280.

[30] Yuan Z, Felice M. Constrained grammatical error correction using statistical machine translation. In *Proc. the 17th Conf. Computational Natural Language Learning*: *Shared Task*, August 2013, pp.52-61.

Zhuo-Ran Liu is an undergraduate student of School of Software at Beihang University, Beijing. This work was done while he was visiting the State Key Laboratory of Intelligent Technology and Systems at Tsinghua University, Beijing.

Yang Liu is an associate professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. He received his Ph.D. degree in computer application technology from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2007. His research areas include natural language processing and machine translation.