LSBert: Lexical Simplification Based on BERT

Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, Yang Shi, and Xindong Wu, *Fellow, IEEE*

***Abstract*—Lexical simplification (LS) aims at replacing complex words with simpler alternatives. LS commonly consists of three main steps: complex word identification, substitute generation, and substitute ranking. Existing LS methods focus on the contextual information of the complex word in the last step (substitute ranking). However, they miss out the following two facts: (1) The word complexity of a polysemous word is very closely related to its context; (2) The step of substitute generation regardless of the context will inevitably produce a large number of spurious candidates. Therefore, we propose a novel LS system LSBert based on pretrained language model BERT to address the aforementioned issues, which is capable of making use of the wider context when both identifying the words in need of simplification and generating substitute candidates for the complex words. Specifically, LSBert consists of a network for complex word identification by fine-tuning BERT and a network for substitute generation based on BERT. Experimental results show that LSBert performs well in both complex word identification and substitute generation, achieving state-of-the-art results in three benchmarks. To facilitate reproducibility, the code of the LSBert system is available at https://github.com/qiang2100/BERT-LS.**

***Index Terms*—Lexical simplification, BERT, Complex word identification, Sentence simplification.**

I. Introduction

LEXICAL Simplification (LS) is the process of replacing a target word in a sentence with simpler alternatives of equivalent meaning, which is useful for many natural language processing tasks like text simplification and paraphrase generation [1], [2]. LS is an effective way of simplifying a text because some work shows that those who are familiar with the vocabulary of a text can often understand its meaning even if the grammatical constructs used are confusing to them. The LS system is commonly framed as a pipeline of three main steps: Complex Word Identification (CWI), Substitute Generation (SG), and Substitute Ranking (SR). CWI is often treated as an independent task [3]. Existing LS methods mainly focus on the two steps: SG and SR [4], [5]. In this paper, we aim to construct a complete system including CWI, SG, and SR. We will discuss these two issues separately.

To solve the problem of CWI, there is a specialized competition called CWI 2018 shared task [6], in which the best teams generally adopted some ensemble-based techniques, such as CAMB provided by Gooding and Kochmar [7] based on Random Forest. These ensemble-based methods employ a large number of features to capture the complexity of a word. However, they are unable to consider the context of the target word when identifying complex words, thus failing to predict word complexity for polysemous words as well as words in various metaphorical or novel contexts. For example, "molars" is considered as a complex word in "Elephants have four molars..., " but it is considered as a simple word in "... new molars emerge in the back of the mouth". Because there are many familiar words around "molars" in the second sentence, the meaning of "molars" can be inferred from these familiar words. To consider the context of the target word, one recent method [8] employs a bi-directional LSTM, treating CWI as a sequence labeling task. However, this method is worse than CAMB, because existing training datasets for the CWI task only contain a few hundred samples that are not enough to train a good neural network modeling. In most cases, pretrained language models have been employed as encoders for sentence-level natural language processing problems involving various classification tasks [9]. In this paper, we will examine the influence of the pretrained language model on CWI.

The popular LS systems still predominantly use a set of rules for replacing complex words with their frequent synonyms from carefully handcrafted databases (e.g., WordNet) [10] or automatically induced from comparable corpora [11] or paraphrase database [12]. The following work utilizes word embedding models to extract substitute candidates for complex words. Given a complex word, the top high similarity words are selected as substitute candidates from the word embedding modeling whose vectors are closer in terms of cosine similarity with the complex word [3], [13], [14]. Recently, the LS system REC-LS attempts to generate substitute candidates by combining linguistic databases and word embedding models. The above methods consider the contextual information on the subsequent step (SR) to decide whether the substitute candidates are suitable for the context of the complex word. This will bring a serious problem. If substitute candidates generated by these methods contain a large number of spurious candidates, these

Manuscript received February 24, 2021; revised August 24, 2021; accepted September 3, 2021. Date of publication September 9, 2021; date of current version October 6, 2021. This work was supported in part by the National Natural Science Foundation of China under Grants 62076217 and 61906060, and in part by the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT) of the Ministry of Education, China, under Grant IRT17R32. This manuscript is an extended version of the conference paper, titled Lexical Simplification with Pretrained Encoders, published in the 34th AAAI Conference on Artificial Intelligence (AAAI), New York, February 7-12, 2020. The associate editor coordinating the reviewof this manuscript and approving it for publication was Dr. Jianfeng Gao.

Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, and Yang Shi are with the Department of Computer Science, Yangzhou, Jiangsu, China (e-mail: qjp2100@gmail.com; liyun@yzu.edu.cn; qjp2100@163.com; yhyuan@yzu.edu.cn; Shiy@yzu.edu.cn).

Xindong Wu is with the Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei, Anhui 230009, China, and also with the Mininglamp Academy of Sciences, Minininglamp, Beijing 100864, China (e-mail: xwu@hfut.edu.cn).

Digital Object Identifier 10.1109/TASLP.2021.3111589

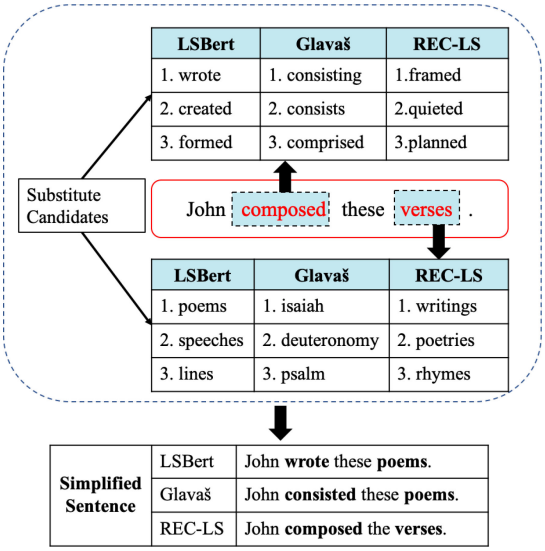

Fig. 1. Comparison of substitute candidates of complex words. Given one sentence "John composed these verses." and complex words 'composed' and 'verses,' the top three simplification candidates for each complex word are generated by our method LSBert and the state-of-the-art two baselines (Glavaš [13] and REC-LS [4]). The simplified sentences by the three LS methods are shown at the bottom.

candidates will seriously confuse the SR step of the LS method. For example, in an extreme case, if simpler alternatives of the complex word do not exist in substitute candidates, the SR step of LS is meaningless.

Therefore, the context of the complex word also plays a central role in fulfilling the step of substitute generation. In contrast to the existing LS methods that only consider the context in the last step (SR), we present a novel complete LS system LSBert, which incorporates the context into each step of LS system. As word complexity depends on context, LSBert adopts a sequence labeling method to identify complex words by fine-tuning pre-trained language modeling BERT [9]. To produce a suitable substitute for the complex word, we also explore the potential of BERT. More specifically, we mask the complex word w of the original sentence S as a new sentence S', and concatenate the original sequence S and S' for feeding into BERT to obtain the probability distribution of the vocabulary corresponding to the masked word. Then we choose the top probability words as substitute candidates. LSBert simplifies one word at a time and is recursively applied to simplify the sentence by taking word complexity in context into account.

Here, we give a simple example shown in Fig. 1. For complex words 'composed' and 'verses' in the sentence "John composed these verses.," the top three substitute candidates of the two complex words generated by the state-of-the-art LS systems [4], [13] are only related with the complex words itself regardless of the context. For example, the candidates "consisting, consists, comprised" are generated by Glavaš [13] for the complex word "composed," and the candidates "framed, quieted, planned" are produced by REC-LS [4]. After choosing the top 10 candidates and matching the POS-tag of the original word, the simplified sentence generated by Glavaš is "John consisted these poems". The simplified sentence generated by REC-LS is "John composed the verses," because it prefers to save the original words when the substitutes cannot fit for the sentence. We can see that the meaning of the original sentence simplified by Glavaš is changed, and REC-LS does not make the right simplification. LSBert generates the appropriate substitutes and achieves its aim that replaces complex words with simpler alternatives.

In summary, the major contributions of this paper are:

1) We propose a novel CWI method by fine-tuning BERT to identify complex words. Experiments results show the effectiveness of our method on three CWI corpora.
2) We propose a novel method that can take full advantage of BERT to generate and rank substitute candidates. To our best knowledge, this is the first attempt to apply BERT for LS. Experimental results show that LSBert obtains obvious improvement compared with the baselines.
3) The proposed LSBert is a simple, effective, and complete LS system: (Simple) many steps used in existing LS systems have been eliminated from our system, e.g., morphological transformation; (Effective) it obtains new state-of-the-art results on three benchmarks; (Complete) LSBert recursively simplifies all complex words in a sentence without requiring additional tools.

The rest of this paper is organized as follows: In Section 2, we introduce the related work of lexical simplification; Section 3 describes the system LSBert; In Section 4, we describe the experimental setup and evaluate the proposed method LSBert; Finally, we draw our conclusions in Section 5.

II. Related Work

Existing complex word identification (CWI) methods can be classified into five classes. (1) Simplify everything methods [9], [15] are the earliest CWI methods, which assume that all words should be simplified. However, it is rarely used now. (2) Threshold-based methods [15], [16] treat these words exceeding the threshold as complex words, in which the commonly used threshold metrics are word frequency or word length. Threshold-based methods are intuitive and easy to implement. However, a single simplicity feature cannot distinguish complex words very well. (3) Lexicon-based methods [10], [18] assume that these words belonging to the lexicon of complex words or simple words are regarded as complex words or simple words. But, constructing the lexicons of complex or simple words is very expensive. (4) Classifier-based methods [6], [7] are the best-performing systems in CWI 2018 shared task, especially ensemble-based classifier methods. However, for the type of methods, there is a huge demand for feature engineering. (5) Sequence labeling-based method [4] treats CWI as a sequence labeling task, and adopts a bi-directional LSTM neural network modeling. It effectively solves the bottleneck of those feature engineering methods. However, the sequence labeling-based method is worse than the ensemble-based method, because existing training datasets only have a few hundred samples that are not enough to train a neural network modeling very well.

Existing lexical simplification (LS) methods mainly focus on substitute generation and substitute ranking, which can be divided into three classes. (1) Linguistic-based methods [12], [19], [20] are the popular LS methods, which generate the

substitute candidates from WordNet or other linguistic databases. However, broad coverage versions of such resources are not available for most languages, and building them is expensive and time-consuming. (2) Rule-based methods [21]–[23] extract paraphrased from articles aligned at document level as possible substitute candidates. For example, some methods [21] are based on the edit history of Simple English Wikipedia compared with English Wikipedia. Pavlick and Nenkova [12] tried to extract rules from PPDB that is a large database of paraphrases extracted from bilingual parallel corpora. The main limitation of rule-based methods relies heavily on parallel corpora. (3) Embedding-based methods [3], [13] based on word embeddings extract the top words as substitute candidates whose vectors are closer in terms of cosine similarity with the complex word. REC-LS [4] attempts to generate substitutes from multiple sources, e.g., WordNet, Big Huge Thesaurus[1] and word embeddings.

After examining existing LS methods ranging from rules-based to embedding-based, the major challenge is that they generate simplification candidates for the complex word regardless of the context of the complex word, which will inevitably produce a large number of spurious candidates that confuse the systems employed in the subsequent steps.

Our method exploits recent advances in BERT [9] to generate suitable simplifications for complex words. Our method generates the candidates of the complex word by considering the whole sentence that is easier to hold cohesion and coherence of a sentence. The previous version was published in artificial intelligence conference (AAAI) [24], which only focused on substitute generations based on BERT. One recent work based on BERT [25] was almost simultaneously proposed with our previous version, which also only focused on substitute generations. Instead of masking the complex word of the input sentence, the proposed method [25] applies the dropout mechanism to the complex word's embeddings for partially masking the word. But the dropout mechanism using BERT does not significantly improve the performance compared with the masking strategy. In this paper, we propose a complete LS system LSBert including complex word identification, substitute generations, and substitute ranking. We mainly add the following four works compared with our previous version: (1) We propose a novel CWI method by fine-tuning BERT. (2) In the steps of substitute generation and substitute ranking, we add an additional strategy to generate substitute candidates by randomly masking a certain percentage of words in the original sentence, and add a new feature paraphrase database (PPDB) to rank the substitutes. (3) LSBert can simplify all complex words of one sentence recursively. (4) We do more experiments to analyze LSBert and the baselines.

III. Lexical Simplification System

In this section, each step of our lexical simplification system LSBert is presented in Fig. 2, which includes the following three steps.

[1][Online]. Available: https://words.bighugelabs.com

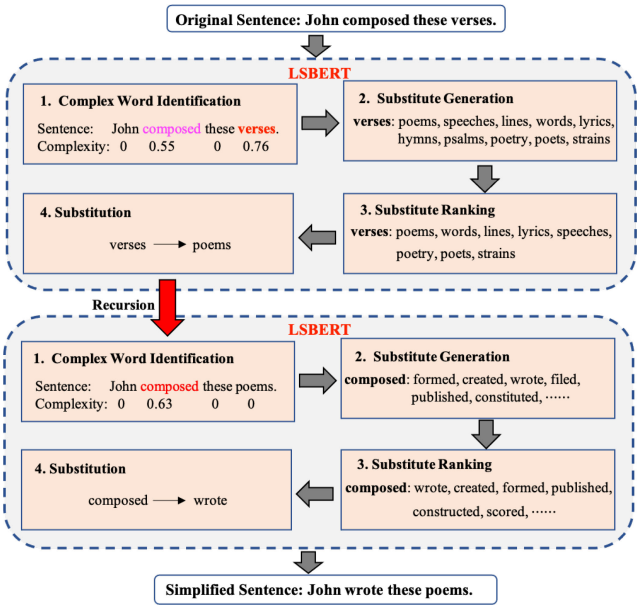

Fig. 2. Overview of the lexical simplification system LSBert.

Complex Word Identification (CWI): Identifying complex words from one sentence has been studied for years, whose goal is to select the words in a given sentence which should be simplified [6], [26]. In our paper, CWI is treated as a sequence labeling task. We fine-tune pretrained language modeling BERT to predict the binary complexity of words. One word is chosen as the complex word when its maximum prediction probability is greater than the pre-defined threshold. To our knowledge, this is the first work that focuses on pretrained language modeling for the CWI task.

Substitute Generation (SG): Giving a sentence S and the complex word w, SG aims to produce the substitute candidates for the complex word w. We replace the complex word with a [MASK] symbol and produce the substitute candidates for the complex word based on BERT. In contrast to existing SG methods, our method makes full use of the context of the complex word.

Substitute Ranking (SR): Giving substitute candidates of the complex word, the SR step is to rank the substitute candidates that fit the context of the complex word. LSBert incorporates Multiple features to rank the substitute candidates. In addition to the language model, similarity, frequency features commonly used in other LS methods, LSBert considers two additional features: probability and PPDB (A Paraphrase Database for Simplification) [12].

After obtaining the ranking of the candidates, we will decide whether the complex word will be replaced by the highest-ranking word by comparing the frequency or language model features of the complex word and the highest-ranking word. Every time LSBert tries to simplify one complex word that owns the maximum prediction probability. After completing one replacement, it will re-identify complex words of the new sentence and recursively simplify another complex word until there is no complex word in the sentence.

A. Fine-Tuning BERT for CWI

Let $S = [w_1, w_2, \ldots, w_n]$ represents a sentence that contains a sequence of words, and $Y = [y_1, y_2, \ldots, y_n]$ represents the labels corresponding to each word, where w_i stands for i-th word in the sentence. CWI can be considered as a task of sequence labeling task by assigning a label $y_i \in \{1,0\}$ for each word, where $\{1,0\}$ indicates whether the word is complex or not.

BERT is a self-supervised method for pre-training a deep transformer encoder, which only requires a large collection of unlabeled text. BERT optimizes two training objectives: masked language modeling (MLM) and next sentence prediction (NSP). The MLM task is used to learn to fill the word at the masked position that is sampled randomly in the input sentence. The NSP task takes two sentences S_1 and S_2 as input, and predicts whether S_2 is the direct continuation of S_1. The two sentences are separated by a special [SEP] token. Additionally, a special [CLS] token is added into the front of S_1 and S_2 to form the input, where the target of [CLS] is whether S_2 indeed follows S_1 in the corpus.

We treat CWI as a sequence labeling task, and fine-tune BERT to predict the word complexity. We first add token [CLS] and [SEP] at the beginning and ending of S. Before feeding S into BERT, we transform it into a sequence of tokens through the WordPiece tokenizer. Specifically, the representation of the top layer in BERT is used as the representation for each token. Because of WordPiece tokenizer, a word may be converted into one or more tokens. We only use the representation of the first token as an input for a classifier to fine-tune BERT.

In neuronal networks tasked with binary classification [4], [6], sigmoid activation in the last layer and binary classification entropy as the loss function are standardized. Here we add a linear layer on the BERT outputs and use a sigmoid function to get the predicted score $\widehat{y}_i$ for each token:

$$\widehat{y}_i = \sigma(W_o \times h_i + b_o) \tag{1}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, h_i represents the vector corresponding to the token w_i, and W_o and b_o are trained parameters.

For each sentence S, the final predicted score $\widehat{Y}$ that is consists of the predicted score $\widehat{y}_i$ of each word by BERT. The loss of the model is the binary classification entropy of prediction $\widehat{Y}$ against gold label Y:

$$loss(S) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log \widehat{y}_i + (1-y_i)\log(1-\widehat{y}_i)) \tag{2}$$

where $loss(S)$ represents the loss of the sentence S.

During the inference phase, the predicted value $\widehat{y}_i$ of each word is obtained using 1. If $\widehat{y}_i$ is greater than 0.5, the word is regarded as complex word; otherwise, the word is regarded as simple word.

For example, the example "John composed$_{0.55}$ these verses$_{0.76}$" is showed in Fig. 2. If the complexity threshold is set to 0.5, the two words "composed" and "verses" will be the complex words.

LSBert first simplifies the word "verses" that owns the highest p value. After completing the simplification process, the complexity of each word in the sentence will be recalculated, excluding words that have been simplified. Besides, we exclude the simplification of entity words by performing named entity identification.

B. Substitute Generation (SG) Based on BERT

Due to the fundamental nature of MLM, MLM is directly used to generate substitutes for complex word. The complex word w in a sentence S is replaced by the special symbol "[MASK]". We can directly mask the complex word w of the sentence S and get the probability distribution of the vocabulary $p(\cdot|S\backslash\{w\})$ corresponding to the masked word w. Finally, we choose the top words from the probability distribution as the substitute candidates.

If we directly let BERT predict the "[MASK]" symbol, BERT is very likely to generate substitute candidates that are semantically different from the complex word although it considers the context. Assume that S replaced by "[MASK]" is denoted as a new sequence S'. Considering BERT is adept at dealing with sentence pairs because of the NSP task, we concatenate the original sequence S and S' as a sentence pair. We feed the sentence pair (S, S') into BERT to predict the "[MASK]" symbol. In such a way, the top words from the predictions $p(\cdot|S, S'\backslash\{w\})$ are related with both the complex word and the context.

In our experiments, the top 10 words from $p(\cdot|S, S'\backslash\{w\})$ are selected as substitute candidates, excluding the morphological derivations of w. Additionally, considering that the contextual information of the complex word is used twice, we randomly mask a certain percentage of words in S excluding w for appropriately reducing the impact of contextual information.

See Fig. 3 for an illustration. Given the sentence "the cat perched on the mat" and the complex word "perched," the top three substitute candidates by our method are "sat, seated, hopped". The top three candidates by Glavaš [13] and REC-LS [4] are "atop, overlooking, precariously," and "put, lighted, lay," respectively. We see that the two methods cannot deal with the problem of polysemy. Our method considering the context can solve the problem very well.

C. Substitute Ranking (SR) Based on Multiple Features

Giving substitute candidates $C = \{c_1, c_2, \ldots, c_k\}$ for the complex word w, we propose a novel SR strategy that incorporates multiple features to rank the candidates, where k is the number of substitute candidates. Previous work for this step is based on the following features: frequency, similarity, and n-gram language modeling, etc. In contrast to previous work, LSBert additionally incorporates two high-quality features: probability and PPDB.

Probability (r_{prob}). The SG step obtains the probability distribution of the vocabulary corresponding to the mask word. Because SG of LSBert already incorporates the contextual information, the probability distribution is a crucial feature that includes the information of both the context and the complex word itself. Therefore, the probability of prediction is treated as

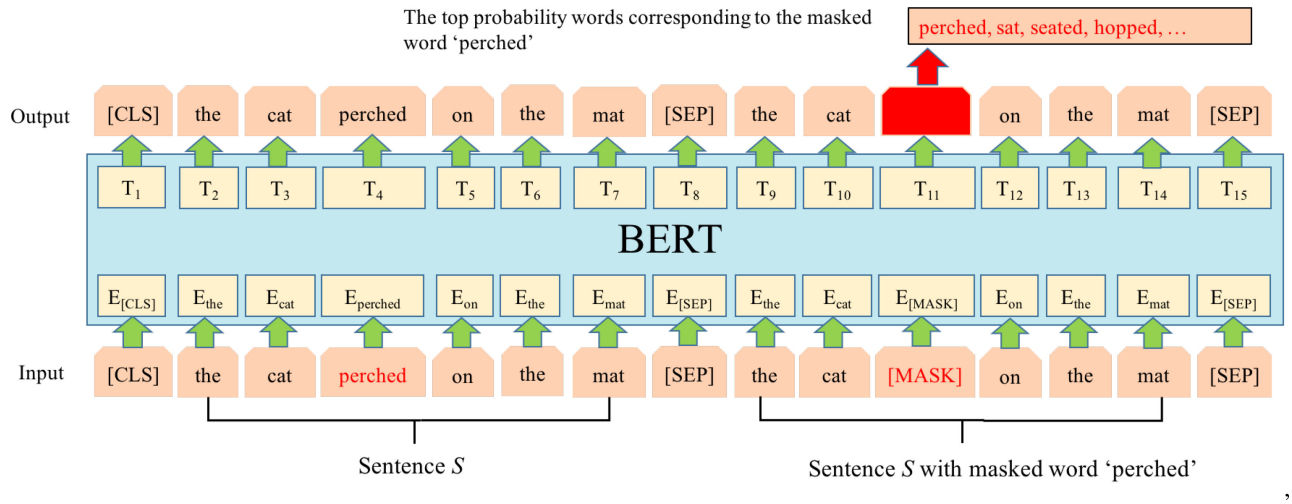

Fig. 3. Substitute generation of LSBert for the complex word prediction, or *cloze* task. The input text is "the cat perched on the mat" with complex word "perched". [MASK], [CLS] and [SEP] are three special symbols in BERT, where [MASK] is used to mask the word, [CLS] is added in front of each input instance and [SEP] is a special separator token.

a ranking feature. If the probability of one candidate is higher, it will have a higher ranking.

Frequency (r_{fre}). Frequency-based candidate ranking strategy [13] is one of the most popular choices by lexical simplification and is quite effective. In general, the more frequency a word is used, the most familiar it is to readers. We adopt the Zipf scale created from the SUBTLEX lists [27], because some experiments [15] revealed that word frequencies from this corpus correlate with human judgments on simplicity than many other more widely used corpora, such as Wikipedia. SUBTLEX[2] is composed of over six million sentences extracted from subtitles of assorted movies. The Zipf frequency of a word is the base-10 logarithm of the number of times it appears per billion words.

Language model feature (r_{lang}). The feature is used to evaluate the fluency of one substitute in a given sentence. Instead of traditional n-gram language modeling, we choose BERT to compute the probability of a sentence or sequence of words. The likelihood of a sentence cannot be directly computed, because BERT is a non-Autoregressive language modeling. We adopt a novel strategy to compute the likelihood of a sentence by BERT.

We use a symmetric window of size g around the complex word as the context of the complex word. Let $W = w_{-g}, \ldots, w_{-1}, w, w_1, \ldots, w_g$ be the context of the original word w. In our experiments, G is set to 5. We first replace the original word w with the substitute candidate c. We then mask each word w_i of W from front to back and feed into BERT to compute the cross-entropy loss of the mask word using 3.

$$loss(w_i) = -\sum_{v_i \in V} \mathbb{I}\{v_i = w_i\} \times \log p_{BERT}(v_i = w_i | W_{\setminus w_i}) \tag{3}$$

where V is the set of words in the vocabulary, $\mathbb{I}\{\cdot\}$ is the indicator function, and p_{BERT} is the BERT prediction distribution (conditioned on the W excluding word w_i).

The language loss of the sequence W is the average of all words,

$$loss(W) = \frac{1}{2g+1} \sum_{i=-g}^{i=g} loss(w_i) \tag{4}$$

Finally, all substitute candidates are ranked based on the corresponding sequence loss $loss(W)$. If the loss of the candidate is lower, it will have a better ranking.

Similarity (r_{sim}). The similarity between the complex word and the substitute candidate is widely used as a feature for SR. In general, word embedding models are used to obtain the vector representation and the cosine similarity metric is chosen to compute the similarity. Assume that the vector representations of the complex word w and the substitute c are v_w and v_c, respectively. The similarity value using cosine is computed by,

$$cos(v_c, v_w) = \frac{\sum_{j=1}^{g} v_c^j v_w^j}{\sqrt{\sum_{j=1}^{g} (v_c^j)^2} \times \sqrt{\sum_{j=1}^{g} (v_w^j)^2}} \tag{5}$$

where g is the dimension of the vector in word embedding modeling and v_c^j is the j-th dimension of the vector representation v_w of word w.

Here, we choose the fastText modeling[3] as the pretrained word embedding modeling. If the similarity value between the candidate and the complex word is greater, the candidate will have a higher ranking.

PPDB (r_{pp}). Some LS methods generate substitute candidates from PPDB or its subset SimplePPDB [28]. In contrast to existing work, we use PPDB as a feature to rank the substitute candidates. PPDB is a collection of more than 100 million

[2][Online]. Available: http://subtlexus.lexique.org

[3][Online]. Available: https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-300d-2M-subword.zip

English paraphrase pairs [29]. These pairs were extracted by using a bilingual pivoting technique, which assumes that two English phrases that translate to the same foreign phrase have the same meaning. We adopt a simple strategy for PPDB to rank the candidates. We hope that the rankings of the substitutes in PPDB and the substitutes not in PPDB maintain a certain gap. If the ranking gap in these two cases is a large value, the influence of other ranking features will be reduced. If the ranking gap is a small value, the influence of this feature will be reduced. Therefore, for each candidate c_i in C, the ranking of c_i is 1 if the pair (w, c_i) exists in PPDB. Otherwise, the ranking number of c_i is $k/3$, where k is the number of words in C.

Each of the features captures one aspect of the suitability of the substitute candidates. LSBert computes five different rankings ($r_{fre}, r_{sim}, r_{lang}$, r_{prob}, and r_{pp}) according to their scores for all substitutes, respectively. The final ranking for all substitutes is computed as follows,

$$f\_r = \lambda_1 r_{fre} + \lambda_2 r_{prob} + \lambda_3 r_{lang} + \lambda_4 r_{sim} + \lambda_5 r_{pp} \quad (6)$$

where f_r denotes the final ranking of C, the weights $\lambda_1, \lambda_2, \lambda_3$, λ_4 and λ_5 balance the relative importance of the different features, and $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 = 1$.

Finally, we choose the substitute with the highest ranking as the top substitute for the complex word.

D. LSBert Algorithm

Following CWI, substitute generation, and substitute ranking steps, the overall simplification algorithm LSBert is shown in Algorithm 1 and Algorithm 2. Given the sentence S and complexity threshold t, we first identify named entity by using an entity identification system.[4] We add the identified entities into $ignore\_list$, which means these words do not need to be simplified.

In the CWI step of LSBert, we calculate the predicted scores of all words in S excluding $ignore\_list$ (line 1). If the number of complex words in the sentence s is larger than 0 (line 2), LSBert will simplify the complex word w owning the high prediction probability (line 3). LSBert calls substitute generation (line 4) and substitute ranking (line 5) in turn. LSBert chooses the top substitute (line 6). One important thing to notice is whether LSBert performs the simplification only if the top substitute has a higher frequency (frequency feature) or lower loss (language modeling feature) than the original word (line 7). Here, we choose the two features (frequency and language modeling) that are calculated by statistical information from large corpora, because the substitute with higher frequency or lower language loss means that it has been well familiar to most people.

When LSBert performs the simplification, it will replace w with top (line 8) and add the word top into $ignore\_list$ (line 9). After completing the simplification of one word, we will iteratively call LSBert (line 10 and line 12). If the number of complex words in S equals 0, we will stop calling LSBert (line 15).

[4][Online]. Available: https://spacy.io/

Algorithm 1: Lexical Simplification Framework.

Input: Setence (S) and Complexity_threshold (t)
Output: Simplified sentence (SS)
1: $ignore\_list \leftarrow$ Named _ Entity_Identification(S)
2: $SS \leftarrow$ LSBert($S, t, ignore\_list$)

Algorithm 2: LSBert ($S, t, ignore\_list$).

1: $complex\_words \leftarrow$ CWI(S, t)-$ignore\_list$
2: **if** number($complex\_words$) >0 **then**
3: $w \leftarrow$ head($complex\_words$)
4: $subs \leftarrow$ Substitute _ Generation(S, w)
5: $subs \leftarrow$ Substitute _ Ranking($subs$)
6: $top \leftarrow$ head($subs$)
7: **if** $fre(top) > fre(w)$ or $loss(top) < loss(w)$ **then**
8: $SS \leftarrow$ Replace(S, w, top)
9: $ignore\_list$.add(w)
10: LSBert($SS, t, ignore\_list$)
11: **else**
12: LSBert($S, t, ignore\_list$)
13: **end if**
14: **else**
15: return S
16: **end if**

IV. Experiments

We evaluate each step of LSBert, and design experiments to answer the following questions:

Q1. The effectiveness of CWI: Does the proposed CWI approach of LSBert outperforms the baselines?

Q2. The effectiveness of substitute generation: Does the substitute generation of LSBert outperform the state-of-the-art competitors?

Q3. The effectiveness of the LS system: Does the effectiveness of LSBert outperforms the full pipeline of the state-of-the-art competitors?

A. Experiment Setup

Baselines. We choose the following baselines to comparison.

(1) Linguistic-based methods. **Devlin** [10] extracts synonyms of complex words from WordNet. **Yamamoto** [18] is proposed for Japanese based on dictionary definitions to extract substitute candidates. Here, Yamamoto is adapted for English by using the Merriam Dictionary to extract definitions of complex words.

(2) Rule-based methods. **Biran** [21] and **Horn** [23] perform SG through parallel corpora EW and SEW. **SimplePPDB** [12] performs SG with a filtered paraphrase database (PPDB).

(3) Embedding-based methods. **Glavaš** [13] performs SG with typical word embeddings. **Paetzold-CA** [14] performs SG with context-aware word embeddigns. **PaetzoldNE** [3] performs SG with parallel corpora and context-aware word embeddings. **REC-LS** [4] performs SG with typical word embeddings and linguistic databases.

(4) BERT-based methods. Here, we give multiple strategies to perform SG by using BERT. **BERT-mask**: we directly mask the complex word of the sentence and feed it into BERT. **BERT**: we directly feed the original sentence into BERT to generate substitute generates. **BERT-dropout** [25] applies the dropout mechanism to the complex word's embeddings for partially masking the word. Compared with our proposed methods, the input of the BERT-based methods is based on a single sentence.

(5) Our proposed methods. **LSBert**$_{pre}$ is our previous conference version [24]. **LSBert** is the proposed method in this paper. Both LSBert$_{pre}$ and LSBert feed two sentences into BERT. For SG step in LSBert, we randomly mask 50% of words in S excluding w. In LSBert, $\lambda_1, \lambda_2, \lambda_3$, λ_4 and λ_5 are set to 0.2.

The experimental results of Devlin, Yamamoto, Biran, Horn, and SimplePPDB, Glavaš, Paetzold-CA, and Paetzold-NE are from these two papers [3], [15]. For the REC-LS method, we use the code proposed by the authors. **BERT-dropout** was re-implemented based on the original paper. In all experiments for BERT-based methods, we use BERT-Large, Uncased (Whole Word Masking) pretrained on BooksCorpus and English Wikipedia.[5]

Dataset. We choose the following datasets to evaluate LSBert on three LS datasets and one text simplification dataset.

(1) We use three widely used lexical simplification datasets (LexMTurk[6] [23], BenchLS[7] [14], NNSeval[8] [15]) to do experiments. The details of the three datasets are illustrated in this paper [15]. Notice that, because these datasets already offer the target words regarded as complex by human annotators, we do not address complex word identification task in our evaluations using the three datasets. These datasets contain instances composed of a sentence, a target complex word, and a set of suitable substitutes provided and ranked by humans for their simplicity.

(2) We use one widely used text simplification dataset (WikiLarge) to simplify one sentence [30]. The training/development/test sets in WikiLarge have 296,402/2000/359 sentence pairs, respectively. WikiLarge is a set of automatically aligned complex-simple sentence pairs from English Wikipedia (EW) and Simple English Wikipedia (SEW). Its validation and test sets are taken from Turkcorpus, where each original sentence has 8 human simplifications created by Amazon Mechanical Turk workers.

B. Evaluation of CWI

Because CWI is treated as an independent task, we will choose the following benchmark and metric to evaluate.

Datasets: We use the English part of the CWI 2018 shared task [6] that contains three datasets: News, WikiNews and Wikipedia. The number of training, validation, and test sets in the three datasets are 946/128/172, 651/85/105, and 388/53/61, respectively.

[5][Online]. Available: https://github.com/google-research/bert

[6][Online]. Available: http://www.cs.pomona.edu/ dkauchak/simplification/lex.mturk.14

[7][Online]. Available: http://ghpaetzold.github.io/data/BenchLS.zip

[8][Online]. Available: http://ghpaetzold.github.io/data/NNSeval.zip

TABLE I
EVALUATION RESULTS OF DIFFERENT CWI METHODS USING F1 METRIC

	News	WikiNews	Wikipedia
CAMB	0.8633	0.8371	0.7780
SEQ	0.8592	0.8060	07634
LSBert	**0.8762**	**0.8595**	**0.8491**

Baselines: For comparison, we choose two state-of-the-art baselines. (1) CAMB [7] based on random forest achieves the best result in CWI 2018 shared task. (2) SEQ [8] employs a bi-directional LSTM, treating CWI as a sequence labeling task. We use the code provided by the authors. (3) Our method (LSBert): uses the BERT-base architecture which consists of 12 self-attention layers, and is fine-tuned on CWI 2018 shared task. Adam with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-7, L_2$ weight decay of 0 is used for fine-tuning.

Metric: The evaluation metric is the macro-averaged F1, which is used in the 2018 CWI shared task.

The results are presented in Table I. Our method LSBert achieves the best results compared with the two baselines on all three datasets. Additionally, LSBert is simpler than CAMB, because CAMB relies on 27 manual features and LSBert need not supply any feature. Although SEQ also considers the context of the target word, CAMB and LSBert outperform SEQ, because the training instances of three datasets are far from enough to meet the requirements of a neural network used by SEQ. Combining the training instances of the three datasets as a whole training dataset to train SEQ and LSBert, SEQ achieves 0.8763, 0.8540 and 0.8140 values, and LSBert achieves 0.8776, 0.8795 and 0.8528 values. We see that SEQ and LSBert are more effective than those trained on a single training dataset.

C. Quantitative Evaluation

(1) Evaluation of Substitute Candidates

The following three widely used metrics are chosen for evaluation [14], [15], [31]. Suppose that there are m samples in the test set, the complex word of the i-th sample is w_i, the set of the annotated substitutes for w_i is p_i, and the set of the generated substitute candidates is q_i. Here, $\#(p_i)$ and $\#(q_i)$ we use are denoted as the number of words in p_i and q_i, respectively.

Precision (PRE): The proportion of generated substitute candidates that are in the annotated substitutes.

$$\text{Precision} = \frac{\sum_{\text{i=1}}^{\text{m}} \#(\text{p}_\text{i} \cap \text{q}_\text{i})}{\sum_{\text{i=1}}^{\text{m}} \#\text{q}_\text{i}} \tag{7}$$

Recall (RE): The proportion of annotated substitutes that are included in the generated substitute candidates.

$$\text{Recall} = \frac{\sum_{\text{i=1}}^{\text{m}} \#(\text{p}_\text{i} \cap \text{q}_\text{i})}{\sum_{\text{i=1}}^{\text{m}} \#\text{p}_\text{i}} \tag{8}$$

F1: The harmonic mean between Precision and Recall.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

The results are shown in Table II. As can be seen, our model LSBert obtains the highest Recall and F1 scores on three datasets, largely outperforming the previous best baseline

TABLE II
EVALUATION RESULTS OF SUBSTITUTE GENERATION ON THREE DATASETS

	LexMTurk			BenchLS			NNSeval		
	PRE	RE	F1	PRE	RE	F1	PRE	RE	F1
Yamamoto	0.056	0.079	0.065	0.032	0.087	0.047	0.026	0.061	0.037
Devlin	0.164	0.092	0.118	0.133	0.153	0.143	0.092	0.093	0.092
Biran	0.153	0.098	0.119	0.130	0.144	0.136	0.084	0.079	0.081
Horn	0.153	0.134	0.143	0.235	0.131	0.168	0.134	0.088	0.106
Glavaš	0.151	0.122	0.135	0.142	0.191	0.163	0.105	0.141	0.121
Paetzold-CA	0.177	0.140	0.156	0.180	0.252	0.210	0.118	0.161	0.136
Paetzold-NE	**0.310**	0.142	0.195	**0.270**	0.209	0.236	0.186	0.136	0.157
REC-LS	0.151	0.154	0.152	0.129	0.246	0.170	0.103	0.155	0.124
BERT-mask	0.254	0.197	0.222	0.176	0.239	0.203	0.138	0.185	0.158
BERT	0.256	0.199	0.224	0.210	0.285	0.242	0.154	0.205	0.176
BERT-dropout	0.255	0.198	0.223	0.204	0.277	0.235	0.153	0.204	0.175
LSBert$_{pre}$	0.287	0.223	0.251	0.231	0.314	0.267	0.185	0.246	0.211
LSBert	0.306	**0.238**	**0.268**	0.244	**0.331**	**0.281**	**0.194**	**0.260**	**0.222**

Paetzold-NE, increasing 37.4%, 19.1%, and 41.4% using the F1 metric. The baseline Paetzold-NE by combining the Newsela parallel corpus and context-aware word embeddings obtains better results on PRE than LSBert, because it uses a different calculation method. If one candidate exists in the gold standard, different morphological derivations of the candidate in substitute candidates are all counted into the PRE metric. Because of considering the context, the substitute candidates of BERT-based methods are normally different words.

We note that BERT-based methods are not only able to outperform other systems on all datasets based on F1, but they also have two clear practical advantages: (1) the only input information it uses at run time is BERT without requiring linguistic database and comparable corpus, (2) the substitute candidates based on BERT do not require additional morphological transformation for the replaced word.

For these baselines based on a single sentence (BERT-mask, BERT and BERT-dropout), the gap between them is very small. Compared with BERT based on a single sentence, our method LSBert$_{pre}$ and LSBert have better results, which verify that our strategy based on sentence pairs is more suitable for lexical simplification. Compared with LSBert$_{pre}$, LSBert randomly masks a certain percentage of words in S. LSBert outperforms LSBert$_{pre}$, which verifies that the strategy is useful for LSBert. In conclusion, the results clearly show that LSBert provides a good balance of precision and recall.

(2) Evaluation of SG and SR

In this section, we evaluate the performance of various LS systems combining SG and SR. We adopt the following two well-known metrics used by these work [15], [23].

Suppose that there are m samples in the test set, the complex word of the i-th sample is w_i, the set of the annotated substitutes for w_i is p_i, and the replacement of the original word is t_i.

Precision (PR): The proportion with which the replacement of the original word is either the original word itself or is in the gold standard.

$$\text{Precision} = \frac{\sum_{\text{i}=1}^{\text{m}}(\mathbb{I}\{\text{t}_\text{i} = \text{w}_\text{i}\} \| \mathbb{I}\{\text{t}_\text{i} \in \text{p}_\text{i}\})}{\text{m}} \tag{10}$$

where, if t_i and w_i are the same word, $\mathbb{I}\{t_i = w_i\}$ is set to 1, else 0; if t_i belonging to p_i is set to 1, else 0.

TABLE III
THE EVALUATION RESULTS USING PRECISION (PR) AND ACCURACY (ACC) ON THREE DATASETS

	LexMTurk		BenchLS		NNSeval	
	PR	ACC	PR	ACC	PR	ACC
Yamamoto	0.066	0.066	0.044	0.041	0.444	0.025
Biran	0.714	0.034	0.124	0.123	0.121	0.121
Devlin	0.368	0.366	0.309	0.307	0.335	0.117
PaetzoldCA	0.578	0.396	0.423	0.423	0.297	0.297
Horn	0.761	0.663	0.546	0.341	0.364	0.172
Glavaš	0.710	0.682	0.480	0.252	0.456	0.197
PaetzoldNE	0.676	0.676	0.642	0.434	0.544	0.335
REC-LS	0.786	0.256	**0.734**	0.335	**0.665**	0.218
LSBert$_{pre}$	0.770	0.770	0.604	0.604	0.420	0.420
LSBert	**0.864**	**0.792**	0.697	**0.616**	0.526	**0.436**

Accuracy (ACC): The proportion with which the replacement of the original word is not the original word and is in the gold standard.

$$\text{Accuracy} = \frac{\sum_{\text{i}=1}^{\text{m}} \mathbb{I}\{\text{t}_\text{i} \in \text{p}_\text{i}\}}{\text{m}} \tag{11}$$

It can be seen from these two metrics that if no simplification is carried out, the PR value is 1 and the ACC value is 0. If all complex words are replaced by substitutes, the PRE and ACC values have the same value.

The results are shown in Table III. Our method LSBert attains the highest accuracy on three datasets, which has an average increase of 29.8% over the former state-of-the-art baseline (Paetzold-NE). It suggests that LSBert is the most proficient in promoting simplicity. Paetzold-NE obtains higher than LSBert on Precision on NNSeval, which also means that many complex words are replaced by the original word itself, due to the shortage of simplification rules in parallel corpora. REC-LS achieves the best PRE and poor ACC, because it prefers the original word as the substitute word.

In conclusion, although LSBert only uses raw text for pre-trained BERT without using any resources, LSBert remains the best lexical simplification method. The results are in accordance with the conclusions of the results of substitute generation.

(3) Evaluation of LS System for Sentence Simplification

The evaluation of LS needs to be provided with the sentence and the specified complex word. Here, we try to simplify one

TABLE IV
COMPARISON OF TEXT SIMPLIFICATION METHODS ON WIKILARGE DATASET

	Methods	SARI	FRES
TS methods	DRESS-LS (2017)	37.27	75.33
	EditNTS (2019)	38.22	73.81
	PBMT (2019)	39.08	76.50
	Access (2019)	**41.87**	**81.55**
LS methods	Glavaš	30.70	**81.82**
	REC-LS	37.11	69.58
	LSBert	**39.37**	77.07

sentence instead of one word of one sentence, and choose a sentence simplification dataset (WikiLarge) for evaluation.

Since most LS methods only focused on one or two steps (SG or SR) of LS, they cannot directly simplify one sentence. Here, we choose two complete LS systems (Glavaš [13] and REC-LS [4]) to comparison. In additional, we choose four state-of-the-art text simplification (TS) methods DRESS-LS [30], EditNTS [32], PBMT [33], and Access [34] as a reference. The first three TS methods except PBMT are sequence-to-sequence models and all need training data sets to learn. PBMT is an unsupervised text simplification system based on a phrase-based machine translation system. For LS methods, they only use the test set to output the simplified sentences. For LSBert and Rec-LS, the complexity threshold of CWI is 0.5. For Glavaš method [13], it tries to simplify all content words (noun, verb, adjective, or adverb) of one sentence.

Following previous work, two widely used metrics (SARI and FRES) in text simplification are chosen [35], [36]. SARI [30] is a text-simplification metric by comparing the output against the simple and complex simplifications.[9] Flesch reading ease score (FRES) measures the readability of the output [37]. A higher FRES represents the simpler output.

Table IV shows the results of all models on WikiLarge dataset. Our model LSBert obtains a SARI score of 39.37 and a FRES score of 77.07, even outperforming these three supervised TS systems (DRESS-LS, EditNTS, and PBMT), which indicates that the model has indeed learned to simplify the complex sentences. Compared with LS methods Glavaš and REC-LS, LSBert also achieves the best results. The two methods go to two different extremes, in which Glavaš simplifies almost all content words of one sentence and REC-LS prefers to save the original word. On the FRES metric, we see that Glavaš outperforms LSBert, which is also because it simplifies almost all content words without caring for the equivalent meaning with the original sentence. Compared with Access, our model is highly competitive, because LSBert does not need a parallel dataset to learn and only focuses on simplifying the words. In conclusion, we see that LSBert outperforms previous LS baselines, even some supervised TS baselines, which indicates that our method is effective at creating simpler output.

D. Ablation Study of LSBert

To further analyze the advantages and the disadvantages of LSBert, we make an ablation study of LSBert in this subsection.

[9]We used the implementation of SARI in [36].

TABLE V
ABLATION STUDY RESULTS OF THE RANKING FEATURES

	LexMTurk		BenchLS		NNSeval	
	PR	ACC	PR	ACC	PR	ACC
LSBert	**0.864**	**0.792**	0.697	0.616	0.526	**0.436**
w/o Probability	0.828	0.774	0.680	**0.629**	0.456	0.406
w/o Language	0.828	0.744	0.670	0.610	0.527	0.418
w/o Similarity	0.820	0.768	0.659	0.607	0.452	0.397
w/o Frequency	0.842	0.694	**0.713**	0.554	**0.556**	0.393
w/o PPDB	0.852	0.784	0.698	0.622	0.502	0.422

TABLE VI
INFLUENCE OF DIFFERENT BERT MODELS

		SG			SR	
		PRE	RE	F1	PR	ACC
LexMTurk	Base	0.317	0.246	0.277	0.744	0.704
	Large	**0.333**	**0.259**	**0.291**	0.792	0.750
	WWM	0.306	0.238	0.268	**0.864**	**0.792**
BenchLS	Base	0.233	0.317	0.269	0.586	0.537
	Large	**0.252**	**0.342**	**0.290**	0.636	0.589
	WWM	0.244	0.331	0.281	**0.697**	**0.616**
NNSeval	Base	0.172	0.230	0.197	0.393	0.347
	Large	0.185	0.247	0.211	0.402	0.360
	WWM	**0.194**	**0.260**	**0.222**	**0.526**	**0.436**

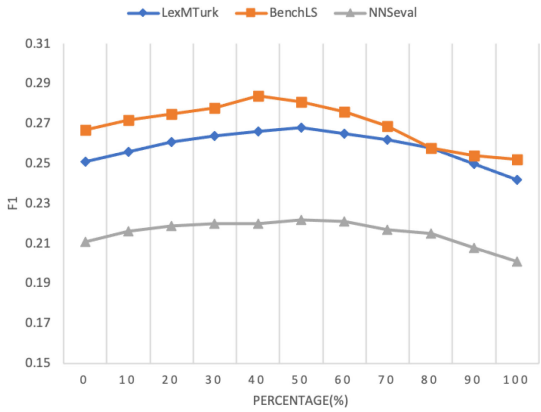

Fig. 4. Influence of the masked percentage of words in S.

(1) Influence of Ranking Features

To determine the importance of each ranking feature, we make an ablation study by removing one feature in turn. The results are presented in Table V. LSBert combining all five features achieves the best results in most cases, which means all features have a positive effect. LSBert removing frequency feature achieves better results on PR metric, but it decreases the values of ACC, because LSBert removing frequency feature prefers to save the original word. These features have different contributions to LSBert's performance. For example, the PPDB feature brings the least impact on the performance of LSBert compared with the other features. In our experiments, all features in LSBert are regarded as equally important, which may not be the best option. In the future, we will try to combine these features by using different weights to further improve the performance of LSBert.

(2) Influence of Different BERT Models for Substitute Generation

The pretrained modeling BERT plays one vital role in LSBert. BERT has different versions based on the parameter scale and training strategy. Here, we attempt to investigate the influence

TABLE VII
THE EXAMPLES OF SUBSTITUTE CANDIDATES THAT DO NOT CONTAIN ONE VALID SUBSTITUTE PROVIDED BY HUMANS ON LEXMTURK. THE COMPLEX WORD OF EACH SENTENCE IS SHOWN IN BOLD

Sent1	Much of the water carried by these streams is **diverted** .
Labels	drawn away, redirected, changed, turned, moved, rerouted, led away, sent away, separated, switched, split, ...
LSBert	reclaimed, displaced, transferred, derived, pumped, routed, converted, recycled, discarded, drained
Sent2	... , every person born into the world is enslaved to the service of sin and , apart from the **efficacious** or prevenient grace of God, ...
Labels	ever, present, showy, useful, effective, capable, strong, valuable, powerful, active, efficient, helpful, generous, power, kindness, effect, ...
LSBert	benevolent, exemplary, abundant, extraordinary, essential, inspired, ubiquitous, irresistible, exclusive, inclusive
Sent3	The Amazon Basin is the part of South America drained by the Amazon River and its **tributaries** .
Labels	streams, branches, riverlets, adjacent, smaller rivers, channels, rivers, brooks, ditches, children creeks, offshoots, creeks
LSBert	basins, drains, derivatives, headwaters, components, subsidiaries, minions, rays, sources, forks
Sent4	He held several **senior** positions in the Royal Flying Corps during World War I, ...
Labels	high-level, older, upper, top, higher, high, superior, important, veteran, head, advance, top-level, advanced, leader, chief, principal, big
LSBert	junior, significant, prestigious, leadership, civil, command, formal, prominent, subordinate, powerful
Sent5	On 1 October 1983 the pilot project began operations as a **full-fledged** bank and was renamed the Grameen Bank to ...
Labels	real, developed, fully operating, legitimate, total, complete, full, qualified, whole, major, working, full-service, ...
LSBert	commercial, development, national, community, central, formal, private, public, bangladeshi, chartered
Sent6	The principal greenhouse , in an art nouveau style with ... , **resembles** the mid-19th century Crystal Palace in London.
Labels	is similar to, looks like, looks-like, mimics, represents, matches, shows, mirrors, echos, look like, favors, appears like, ...
LSBert	recalls, suggests, approaches, echoes, references, parallels, appears, depicts, incorporates, follows
Sent7	A perfectly elastic collision is defined as one in which there is no loss of **kinetic** energy in the collision.
Labels	active, moving, movement, motion, static, motive, innate, kinetic, real, strong, driving, motion related, motion-, living, powerful, ...
LSBert	mechanical, rotational, dynamic, total, thermal, momentum, physical, the, potential, energetic
Sent8	None of your watched items were **edited** in the time period displayed.
Labels	changed, looked at, refined, revise, finished, fixed, revised, revised, scanned, shortened
LSBert	altered, incorporated, appropriate, modified, organized, filtered, included, blended, amended, enhanced

of different BERT versions on the performance of LSBert. We choose the following three BERT models:

(1) BERT-based, uncased (Base): 12-layer, 768-hidden, 12-heads, 110 M parameters.

(2) BERT-large, uncased (Large): 24-layer, 1024-hidden, 16-heads, 340 M parameters.

(3) BERT-large, uncased, Whole Word Masking (WWM): 24-layer, 1024-hidden, 16-heads, 340 M parameters. The above two BERT models randomly select WordPiece tokens to mask. Whole Word Masking always masks all of the tokens corresponding to a word at once.

Table VI shows the results of the experiments based on different BERT models on three datasets. From Table VI, we see that the WWM model obtains the highest accuracy and precision over the two other models. Besides, the Large model outperforms the Base model. It can be concluded that a better BERT model can help to improve the performance of LSBert. If a better BERT model is available in the future, one can try to replace the BERT model in this paper to further improve the performance of the LS system.

(3) Influence of the Masked Percentage of Words in S

Compared with LSBert$_{pre}$, LSBert randomly masks a certain percentage of words in S for appropriately reducing the impact of contextual information in substitute generation (SG). In this subsection, we explore the influence of the masked percentage of words in S for SG. When the masked percentage is zero, the SG of LSBert is the same as the SG of LSBert$_{pre}$. While fixing the other parameters, we set the percentage to vary from 0% to 100%. We select F1 of SG as a metric.

Fig. 4 shows the results on three datasets. For SG step, when increasing the percentage, the score of F1 first increases and then declines. When the percentage is 50%, the performance of LSBert reaches the best results, that is the set percentage of substitute candidates in the paper. Therefore, we can see that the masked percentage of words in S is useful for LSBert.

E. Qualitative Study

All of the above experiments are quantitative analyses of LSBert. Here, we also qualitatively evaluate our system from three aspects: substitute generation, substitute ranking, and sentence simplification.

(1) The Analysis of Substitute Generation Results

When the number of substitute candidates is set to 10, the proportion of LSBert that generates at least one valid substitute candidate is 98.6% on LexMTurk dataset, where the number of instances in Lexmturk is 500. Namely, LSBert only produces no effective substitute in only 8 sentences. When the number of generated candidates is 15, LSBert cannot generate any valid candidates on only 4 sentences. When the number of generated candidates is 30, only one sentence cannot generate a valid candidate by LSBert. In this subsection, we will analyze the 8 sentences when the number of the candidates is set to 10 in Table VII.

We see that LSBert can generate one or two valid substitute candidates on these sentences (sent4, sent5, sent7, and sent8), e.g., "senior->powerful," "full-fledged->development," "kinetic->dynamic," and "edited->altered". Since the labels are provided by humans, it is impossible to provide all suitable substitutes for each word. LSBert fails to produce any valid candidate word on the other sentences (sent1, sent2, sent3, and sent6). When we analyze these wrong substitute candidates, these substitutes can fit the context without concerning equivalent meaning.

(2) The Analysis of Substitute Ranking Results

LSBert finds one or more suitable alternatives for almost all samples, but the final system results do not always select the most suitable candidate as the final substitute. In this subsection, we will analyze the possible reasons for this question. Table VIII shows some examples that LSBert cannot produce the right substitute.

TABLE VIII
THE EXAMPLES THAT THE FINAL SUBSTITUTE GENERATED BY LSBERT IS NOT FROM THE MANUAL LABELS. THE WORDS IN THE SUBSTITUTE RANKING BELONGING TO THE LABELS ARE SHOWN IN BOLD

Sent1	Triangles can also be **classified** according to their internal angles, measured here in degrees.
Labels	grouped, categorized, arranged, labeled, divided, organized, separated, defined, described, ...
LSBert_SR	**divided**, **described**, separated, designated, ...
LSBert_Substitute	classified
Sent2	...; he **retained** the conductorship of the Vienna Philharmonic until 1927.
Labels	kept, held, had, got
LSBert_SR	maintained, **held**, **kept**, remained, continued, shared, ...
LSBert_Substitute	maintained
Sent3	..., and a Venetian in Paris in 1528 also **reported** that she was said to be beautiful
Labels	said, told, stated, wrote, declared, indicated, noted, claimed, announced, mentioned
LSBert_SR	**noted**, confirmed, described, claimed, recorded, **said**, ...
LSBert_Substitute	reported
Sent4	..., the king will **rarely** play an active role in the development of an offensive or ....
Labels	infrequently, hardly, uncommonly, barely, seldom, unlikely, sometimes, not, seldomly, ...
LSBert_SR	never, usually, **seldom**, **not**, **barely**, **hardly**, ...
LSBert_Substitute	never

TABLE IX
THE SIMPLIFIED SENTENCES BY THREE DIFFERENT LS METHODS ON WIKILARGE DATASET. SUBSTITUTES ARE SHOWN IN BOLD

1	Sent1	Admission to Tsinghua is exceedingly competitive.
	Label	Entrance to Tsinghua is very very difficult.
	Glavaš	**Offers** to **Qinghua** is **very exciting**.
	REC-LS	Admission to Tsinghua is exceedingly competitive.
	LSBert	**Entrance** to Tsinghua is **very tough**.
2	Sent2	Many species had vanished by the end of the nineteenth century, with European settlement.
	Label	With European settlement many species have been vanished.
	Glavaš	**Some birds** was **gone** by the **time** of the **twentieth history**, with **world land**.
	REC-LS	Many species had **disappeared** by the end of the **19th** century, with European settlement.
	LSBert	Many **animals** had **disappeared** by the end of the nineteenth century, with European settlement.
3	Sent3	In 1987 Wexler was inducted into the Rock and Roll Hall of Fame.
	Label	In 1987 Wexler was inducted into the Rock and Roll Hall of Fame.
	Glavaš	In 1987 **Livingston** was **fame** into the rock and **you** hall of **hall**.
	REC-LS	In 1987 Wexler was inducted into the Rock and Roll Hall of Fame.
	LSBert	In 1987 Wexler was **elected** into the Rock and Roll Hall of **Honor**.
4	Sent4	Oregano is an indispensable ingredient in Greek cuisine.
	Label	Oregano is a necessary ingredient in Greek cuisine.
	Glavaš	**Garlic** is an **essential** ingredient in Greek **cooking**.
	REC-LS	Oregano is an **essential element** in Greek cuisine.
	LSBert	Oregano is an **important element** in Greek **food**.
5	Sent5	Their eyes are quite small, and their visual acuity is poor.
	Label	Their eyes are quite small, and their visual acuity is poor.
	Glavaš	Their eyes **have very little**, and their **musical visual** is **bad**.
	REC-LS	Their eyes are quite small, and their **ocular acuteness** is poor.
	LSBert	Their eyes are quite small, and their visual **ability** is **bad**.

From the two sentences sent1 and sent3, we observe that the SR step selects the best substitute, but LSBert still chooses the original word. This is because the Zipf value of "divided" is 3.65 and the Zipf value of "classified" is 3.83, where the Zipf value is from the frequency feature in SR. LSBert regards "classified" as a simpler word compared with "divided". It is the same reason for sent3 in which the Zipf value of "noted" is 3.68 and the Zipf value of "reported" is 4.18. Consequently, in sent1 and sent3, the best substitutes of SR cannot be used as the final substitutes.

The second case is that the best substitute for the SR step is not from the labels provided by humans. In sent2 and sent4, LSBert chooses "maintained" as a simpler for "retained" and "never" as a simpler for "rarely". We can find that the words "maintained" and "never" are also suitable substitutes, but do not appear in the labels. These experimental results suggest that LSBert achieves better results that are not reflected in the results of Table II.

(3) The Analysis of Sentence Simplification Results

The above qualitative study for LSBert needs to provide the complex word by humans. In this experiment, we try to verify the results of LS methods on sentence simplification. We also choose the two methods Glavaš and REC-LS to make a comparison. Table IX shows some simplified examples from the WikiLarge dataset. We draw the same conclusions from these examples with the LS system for sentence simplification (See Table IV). Glavaš tries to simplify every content word in the sentence ignoring the aim of LS. LS aims to replace complex words in a given sentence with simpler alternatives of equivalent meaning. Rec-LS can make the right simplifications, e.g., sentence 2. But, for sentence 1 and sentence 3, Rec-LS outputs the original sentence. LSBert replaces complex words with simpler alternatives and makes the most reasonable simplification. It verifies that LSBert is more fit for the LS task than the baselines.

V. Conclusion

We propose a novel BERT-based system LSBert for lexical simplification (LS). The existing LS methods only consider the context of the complex word on the last step (substitute ranking) of LS. LSBert focuses on the context of the complex word on all three steps of lexical simplification. To our best knowledge, LSBert is the first work to utilize BERT during the steps of complex word identification and substitute generation. Experimental results have shown that LSBert achieves the best performance on CWI and LS tasks. Compared with the state-of-the-art text simplification methods, LSBert also gets very good results on WikiLarge dataset. In the future, the pretrained BERT model can be fine-tuned with just a simpler English corpus (e.g., Newsela), and then we will use fine-tuned BERT for lexical simplification.

References

[1] L. Feng, "Automatic readability assessment for people with intellectual disabilities," *ACM Sigaccess Accessibility Comput.*, no. 93, pp. 84–91, 2009.

[2] H. Saggion, "Automatic text simplification," *Synth. Lectures Hum. Lang. Technol.* vol. 10, no. 1, pp. 1–137, 2017.

[3] G. Paetzold and L. Specia, "Lexical simplification with neural ranking," in *Proc. Assoc. Comput. Linguistics: Volume 2, Short Papers*, 2017, pp. 34–40.

[4] S. Gooding and E. Kochmar, "Recursive context-aware lexical simplification," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 4855–4865.

[5] J. Qiang, X. Lv, Y. Li, Y. Yuan, and X. Wu, "Chinese lexical simplification," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 29, pp. 1819–1828, 2021.

[6] S. M. Yimam *et al.*, "A report on the complex word identification shared task 2018," pp. 66–78, 2018.

[7] S. Gooding and E. Kochmar, "Complex word identification with ensemble-based voting," in *Proc. 13th Workshop Innov. Use NLP Building Educ. Appl.*, 2018, pp. 184–194.

[8] S. Gooding and E. Kochmar, "Complex word identification as a sequence labelling task," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1148–1153.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*.

[10] S. Devlin and J. Tait, "The use of a psycholinguistic database in the simpli cation of text for aphasic readers," *Linguistic Databases,* vol. 1, pp. 161–173, 1998 .

[11] J. De Belder, M.-F. Moens, "Text simplification for children," in *Proc. SIGIR Workshop Accessible Search Syst.*, 2010, pp. 19–26.

[12] E. Pavlick and C. Callison-Burch, "Simple PPDB: A paraphrase database for simplification," in *Proc. Assoc. Comput. Linguistics: Volume 2, Short Papers*, 2016, pp. 143–148.

[13] G. Glavaš and S. Štajner, "Simplifying lexical simplification: do we need simplified corpora?," in *Proc. Assoc. Comput. Linguistics*, 2015, pp. 63–68.

[14] G. H. Paetzold and L. Specia, "Unsupervised lexical simplification for non-native speakers," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 3761–3767.

[15] G. H. Paetzold and L. Specia, "A survey on lexical simplification," in *J. Artif. Intell. Res.*, vol. 60, pp. 549–593, 2017.

[16] S. Bott, L. Rello, B. Drndarevic, and H. Saggion, "Can spanish be simpler? lexsis: Lexical simplification for spanish," in *Proc. COLING*, 2012, pp. 357–374.

[17] K. Wróbel, "Plujagh at semeval-2016 task 11: Simple system for complex word identification," in *Proc. 10th SemEval*, 2016, pp. 953–957.

[18] T. Kajiwara, H. Matsumoto, and K. Yamamoto, "Selecting proper lexical paraphrase for children," in *ROCLING*, 2013, pp. 59–73.

[19] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," in *Proc. 5th Annu. Int. Conf. Syst. Documentation*, 1986, pp. 24–26.

[20] M. Maddela and W. Xu, "A word-complexity lexicon and a neural readability ranking model for lexical simplification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3749–3760.

[21] O. Biran, S. Brody, and N. Elhadad, "Putting it simply: a context-aware approach to lexical simplification," in *Proc. Assoc. Comput. Linguistics*, 2011, pp. 496–501.

[22] M. Yatskar, B. Pang, C. Danescu-Niculescu-Mizil, and L. Lee, "For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia," in *NACACL*, Assoc. Comput. Linguistics, 2010, pp. 365–368.

[23] C. Horn, C. Manduca, and D. Kauchak, "Learning a lexical simplifier using wikipedia," in *Proc. Assoc. Comput. Linguistics, Volume 2: Short Papers*, 2014, pp. 458–463.

[24] J. Qiang, Y. Li, Y. Zhu, Y. Yuan, and X. Wu, "Lexical simplification with pretrained encoders," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 8649–8656.

[25] W. Zhou, T. Ge, K. Xu, F. Wei, and M. Zhou, "Bert-based lexical substitution," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3368–3373.

[26] M. Shardlow, "A comparison of techniques to automatically identify complex words," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics Proc. Student Res. Workshop*, 2013, pp. 103–109.

[27] M. Brysbaert and B. New, "Moving beyond kucera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english," *Behav. Res. Methods*, vol. 41, no. 4, pp. 977–990, 2009.

[28] R. Kriz, E. Miltsakaki, M. Apidianaki, and C. Callisonburch, "Simplification using paraphrases and context-based lexical substitution," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 207–217.

[29] J. Ganitkevitch, B. V. Durme, and C. Callison-Burch, "PPDB: The paraphrase database," in *Proc. NAACL-HLT*, 2013, pp. 758–764.

[30] X. Zhang, M. Lapata, "Sentence simplification with deep reinforcement learning," *arXiv:1703.10931*.

[31] G. Paetzold and L. Specia, "Lexenstein: A framework for lexical simplification," in *Proc. ACL-IJCNLP 2015 Syst. Demonstrations*, 2015, pp. 85–90.

[32] Y. Dong, Z. Li, M. Rezagholizadeh, and J. C. K. Cheung, "Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3393–3402.

[33] J. Qiang and X. Wu, "Unsupervised statistical text simplification, *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1802–1806, Apr. 2021.

[34] L. Martin, B. Sagot, and É. de la Clergerie, "A bordes, controllable sentence simplification," *arXiv:1910.02677*.

[35] K. Woodsend and M. Lapata, "Learning to simplify sentences with quasi-synchronous grammar and integer programming," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2011, pp. 409–420.

[36] W. Xu, C. Napoles, E. Pavlick, Q. Chen, and C. Callison-Burch, "Optimizing statistical machine translation for text simplification," *TACL*, vol. 4, pp. 401–415, 2016.

[37] P. J. Kincaid, R. P. Fishburne, R. J. L. Richard, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," Tech. Rep., DTIC Document.

Jipeng Qiang received the Ph.D. degree in computer science and technology from the Hefei University of Technology, China, in 2016. He was the Ph.D. Visiting Student with the Artificial Intelligence Lab, the University of Massachusetts, Boston from 2014 to 2016. He is currently an Associate Professor with the School of Information Engineering, Yangzhou University. His research interests include data mining and natural language processing.

Yun Li received the M.S. degree in computer science and technology from the Hefei University of Technology, China, and the Ph.D. degree in control theory and control engineering from Shanghai University, China, in 1991 and 2005, respectively. He is currently a Professor with the School of Information Engineering, Yangzhou University, China. He has authored or coauthored more than 100 scientific papers. His research interests include data mining and cloud computing.

Yi Zhu received the B.S. degree from Anhui University, China, and the M.S. degree from the University of Science and Technology of China, China, and the Ph.D. degree from the Hefei University of Technology, China. He is currently an Assistant Professor with the School of Information Engineering, Yangzhou University, China. His research interests include data mining and knowledge engineering, and recommendation systems.

Yunhao Yuan received the M. Eng. degree in computer science and technology from Yangzhou University, China, and the Ph.D. degree in pattern recognition and intelligence system from the Nanjing University of Science and Technology, China, in 2009 and 2013, respectively. He is currently an Associate Professor in the School of Information Engineering, Yangzhou University. His research interests include pattern recognition, data mining, and image processing.

Yang Shi received the B.E. degree in network engineering and the M.E. degree in computer science and technology from Chongqing University, Chongqing, China, in 2011 and 2014, respectively, and the Ph.D. degree in information and communication engineering with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, in 2018. He is currently a Lecturer with the School of Information Engineering, Yangzhou University, Yangzhou, China. His main research interests include neural networks, artificial intelligence, and robotics.

Xindong Wu (Fellow, IEEE) received the B.S. and the M.S. degrees in computer science from the Hefei University of Technology, China, and the Ph.D. degree in artificial intelligence from the University of Edinburgh, Britain. He is a Yangtze River Scholar with the School of Computer Science and Information Engineering, the Hefei University of Technology, China, the President of Mininglamp Academy of Sciences, Mininglamp, Beijing, China, and a Fellow of AAAS. His research interests include data mining, Big Data analytics, knowledge-based systems, and Web information exploration. He is currently the Steering Committee Chair of the IEEE International Conference on Data Mining (ICDM), the Editor-in-Chief of Knowledge and Information Systems (KAIS, by Springer), and a Series Editor-in-Chief of the Springer Book Series on Advanced Information and Knowledge Processing (AI&KP). He was the Editor-in-Chief of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE, by the IEEE Computer Society) from 2005 to 2008. He was a Program Committee Chair/Co-Chair of the 2003 IEEE International Conference on Data Mining, the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, and the 19th ACM Conference on Information and Knowledge Management.