An LLM-Enhanced Adversarial Editing System for Lexical Simplification

Keren Tan[1], Kangyang Luo[1], Yunshi Lan[1*], Zheng Yuan[2], Jinlong Shu[1]
[1]School of Data Science & Engineering, East China Normal University, Shanghai, China
[2]Department of Informatics, King's College London, U.K.
{tankeren1020, 52205901003}@stu.ecnu.edu.cn, yslan@dase.ecnu.edu.cn,
zheng.yuan@kcl.ac.uk, jlshu@admin.ecnu.edu.cn

Abstract

Lexical Simplification (LS) aims to simplify text at the lexical level. Existing methods rely heavily on annotated data, making it challenging to apply in low-resource scenarios. In this paper, we propose a novel LS method without parallel corpora. This method employs an Adversarial Editing System with guidance from a confusion loss and an invariance loss to predict lexical edits in the original sentences. Meanwhile, we introduce an innovative LLM-enhanced loss to enable the distillation of knowledge from Large Language Models (LLMs) into a small-size LS system. From that, complex words within sentences are masked and a Difficulty-aware Filling module is crafted to replace masked positions with simpler words. At last, extensive experimental results and analyses on three benchmark LS datasets demonstrate the effectiveness of our proposed method.

Keywords: Lexical simplification, Adversarial editing, Large language models

1. Introduction

Text Simplification (TS) is the process of simplifying a sentence while retaining its semantics as much as possible. It enables to lower the difficulty level of the entire sentence and helps people with cognitive disabilities to understand by making the text more readable (Paetzold and Specia, 2017b). As a special category of TS tasks, Lexical Simplification (LS) restricts the simplification at the lexical level via replacing complex words with alternative simpler words, thus minimising the revision to the original sentences.

Conventional LS tasks broadly consist of two sub-tasks, namely Complex Word Identification (CWI) and Substitute Generation (SG), which focus on detecting complex words and generating alternative words, respectively. So far, a panoply of efforts work on addressing the said LS tasks. For example, early LS systems leverage a set of rules for identifying and substituting complex words with frequent synonyms from external databases (e.g., WordNet (Miller et al., 1990), but these methods suffer from limited flexibility and adaptability (Kajiwara et al., 2013). Recently, popular LS systems first train a model to detect the complex words in a sentence, and then use another model to predict the alternative words, collaborating together to eventually produce a simplified sentence (Qiang et al., 2021; Seneviratne et al., 2022; Wilkens et al., 2022). However, the mentioned two-stage approaches have a heavy reliance on the annotation of CWI and SG sub-tasks, thereby impairing their applications. In real-world scenarios, acquiring parallel corpora for LS tasks is a costly endeavor, not to mention the annotation of CWI and SG. As such, we aim to develop an LS system without parallel corpora in this work, but we are also confronted with the following challenges: (1) In the absence of annotated data, the above-mentioned supervised training approaches are inapplicable, making it considerably challenging to ensure the accuracy of simplification. (2) Constructing the previous two-stage system for LS tasks without parallel corpora is problematic, as in such scenarios, models struggle to learn the transformation from complex sentences to simplified ones.

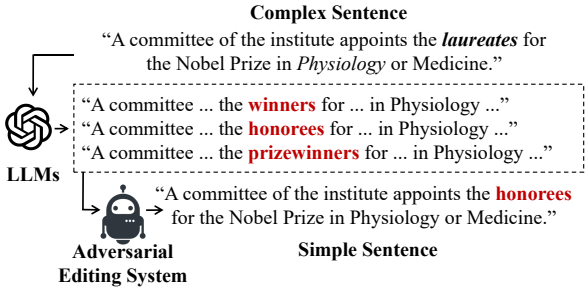

Figure 1: The motivation of our LLM-enhanced Adversarial Editing System, that is, distilling the knowledge from LLMs to our small-size Adversarial Editing System. In the complex sentence, correct complex words are in italic bold fonts, and red bold signifies differences between generated and original sentences.

Inspired by existing study in style transfer (Wang et al., 2022), we develop an Adversarial Editing System to conduct lexical edits to the original sentence with the help of non-parallel corpora, where

* Corresponding author.

LREC-COLING 2024, pages 1136–1146

complex words are masked by the editing system, and the substitutions are generated via a cloze model following the two-stage approaches. Nonetheless, striking a balance between semantic preservation and simplification degree remains a challenging endeavor. For example, in Figure 1, non-stylistic words *"Physiology"* can be accidentally masked and it is hard to distinguish if *"laureates"* is a complex word.

To this end, as the first attempt to LS without parallel corpora, we bring forward a new LS method dubbed as **LAE-LS** (**L**LM-Enhanced **A**dversarial **E**diting System for **L**exical **S**implification), which involves two modules: Adversarial Editing and Difficulty-aware Filling. Concretely, we employ an Adversarial Editing module to train an Edit Predictor for predicting lexical edits, which can be used to identify complex words within sentences. To balance the preservation of semantics and the level of simplification, we introduce a confusion loss and an invariance loss to confuse the discriminator and preserve the semantics of the original sentence, respectively. Particularly, an LLM-enhanced loss is tailored to extract supervision signals from Large Language Models (LLMs) (e.g., ChatGPT) (Brown et al., 2020; Zhang et al., 2022; Touvron et al., 2023). To be specific, we intricately design instructions to guide LLMs in identifying complex words within sentences, rather than directly commanding them to rewrite the original sentences, which can bypass changes to the sentence syntax. With this loss, high-quality knowledge from LLMs can be distilled into our Edit Predictor. To avoid the overwhelming domination of the LLM-enhanced loss, we combine the above losses with weights when training. Eventually, the Difficulty-aware Filling module is crafted to fill in the masked positions with alternative simple words.

In summary, we highlight our contributions as follows:

- We propose a novel LS method LAE-LS that is capable of making lexical edits to the original sentences without parallel corpora, thus rendering it feasible to perform LS tasks in low-resource scenarios.
- To achieving tradeoff between the simplification degree and semantic preservation, we include confusion loss and invariance loss. Furthermore, the LLM-enhanced loss is devised, enabling the distillation from LLMs to a small-size LS system.
- Our method achieves SOTA results on three well-known LS datasets and yields competitive results to `GPT-3.5-turbo` even with a significantly smaller parameter size.

2. Related Work

Lexical Simplification. The early methods for the LS task either resorted to threshold-based strategies (Keskisärkkä and Jönsson, 2012) or relied on dictionaries to identify complex words within sentences (Kajiwara et al., 2013), and took advantage of external resources (e.g., synonym dictionaries (Lesk, 1986), WordNet (Devlin, 1998), word embeddings (Mikolov et al., 2013)) to provide substitutions for complex words. However, these methods suffer from limited flexibility and adaptability (Paetzold and Specia, 2017b). Shortly thereafter, a panoply of modifications for CWI and SG respectively have been proposed to alleviate said issues. For CWI, Gooding and Kochmar (2018) shifts it towards training classifiers using supervised training with significant demand for feature engineering. Recently, some works have achieved significant success by treating the CWI task as a sequence labeling task (Qiang et al., 2021; Gooding and Kochmar, 2019). For SG, the recent methods leverage the contextual comprehension capabilities of pre-trained models to generate substitutions for complex words. For example, LSBert (Qiang et al., 2021) predicts alternative words for complex words by the BERT (Devlin et al., 2019) model. SimpleBART (Sun et al., 2023) augments pre-trained models through fine-tuning, enabling the effectively prediction for simpler words. ParaLS (Qiang et al., 2023) fine-tunes a paraphraser to generate substitute candidates for complex words using two novel decoding strategies. The said methods approach remarkably performance for the LS task, but they rely heavily on supervised training or a substantial amount of external linguistic resources. On this account, our study is inspired by the aforementioned pitfalls.

Adversarial Network and LLM. With the introduction of Generative Adversarial Networks (GAN) (Goodfellow et al., 2020), adversarial learning has become ubiquitous in unsupervised training (Shen et al., 2017; Wang et al., 2022). For example, KiS (Laban et al., 2021) extends Seq2Seq models to unparallel corpora scenarios where the decoder serves as the generator, and a component, which assesses whether a sentence is simplified or not, acts as the discriminator. However, this method cannot guarantee that the syntax of the original sentence remains unchanged. Accordingly, given the outstanding performance of LLMs (Zhang et al., 2022; Brown et al., 2020) in various linguistic tasks (Liang et al., 2023; Lan et al., 2023), a stream of efforts has been explored the use of LLMs in TS, yielding promising results (Feng et al., 2023; Chi et al., 2023; Sun et al., 2023). Nevertheless, due to high resource consumption, time-intensive inference and over-simplification results,

applying LLMs directly to LS is not practical. Yet, we argue that using LLMs as an enhancement tool or component remains a viable option. To the best of our knowledge, how to distil knowledge from LLMs for augmenting LS tasks is unexplored.

3. Method

3.1. Task Definition and Overview

In this paper, we define the LS task, which combines the procedure of CWI and SG, as follows: The goal of LS is to build a system that can convert a complex text X_i labeled by s_x to a simple text Y_i labeled by s_y via replacing certain words with simpler words. Of note, we leverage non-parallel corpora, which is denoted as $\mathcal{D}_x = \{X_i\}$ and $\mathcal{D}_y = \{Y_i\}$, to build an LS system, thereby bypassing parallel corpora.

Our proposed method LAE-LS, which performs lexical simplification, involves two modules: **Adversarial Editing** and **Difficulty-aware Filling**. The overall architecture of our method is displayed in Fig. 2. Specifically, the Adversarial Editing module is an edit-based generative adversarial network designed to make lexical edits, which can be used to mask the complex words in complex sentences. Built upon the well-trained Adversarial Editing module, a Difficulty-aware Filling module is used to fill in the masked position with simple words. Remarkably, unlike the previous filling model (Qiang et al., 2021), the Difficulty-aware Filling module, which is a cloze model, not only considers original sentences as clues but also maintains an awareness of producing simpler words.

3.2. Adversarial Editing

In this subsection, we detail the Adversarial Editing module. Compared to existing efforts (Laban et al., 2021; Zhao et al., 2020; Surya et al., 2019), which directly model the text simplification from X_i to Y_i, our module formulates the lexical simplification as an editing task, the goal of which is to predict lexical edits. As shown in Fig. 2, we aim to train an Edit Predictor to perform the said procedure.

3.2.1. Edit Predictor and Discriminator

Before proceeding formally, we give notations for descriptive convenience as well as present Edit Predictor and Discriminator. We add a special token "[*CLS*]" at the beginning of the sentence and denote a sentence by $X_i = [t_{i,1}, t_{i,2}, ..., t_{i,L}]$, where $t_{i,l}$ ($l \in [L]$) is l-th token in X_i and L is the length of the sentence. For Edit Predictor, given that X_i, its output is $G_i = [g_{i,1}, g_{i,2}, ..., g_{i,L}]$, where $g_{i,l} \in \{K, M\}$ ($l \in [L]$) is the edit operation for $t_{i,l}$ in X_i. Note that "*K*" indicates the current token is not a stylistic token or is already a simple token, and "*M*" indicates the current token should be replaced by a mask token. For example, the edit operation for a complex sentence "[*Much, of, the, water, carried, by, these, streams, is, diverted,* .]" is "[*K*, *K*, *K*, *K*, *K*, *K*, *K*, *K*, *M*, *K*]". In our experiments, the Edit Predictor is built upon a BERT (Devlin et al., 2019) encoder. To be specific, given that X_i, we encode them with a sequence of hidden representations and then predict the edit labels via a Multilayer Perceptron (MLP) as follows:

$$\begin{aligned}
&\mathbf{w}_l = \mathbf{w}_l^{tok} + \mathbf{w}_l^{typ} + \mathbf{w}_l^{pos} (l \in [L]),\\
&\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L] = \text{BERT}(\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_L),\\
&P_G = \text{softmax}(\mathbf{WH} + \mathbf{A}),
\end{aligned} \tag{1}$$

where $\mathbf{w}_l^{tok}$, $\mathbf{w}_l^{typ}$, $\mathbf{w}_l^{pos}$ represent the word embedding, token type embedding, and position embedding of l-th token, respectively. $\mathbf{W}$ and $\mathbf{A}$ are learnable parameters, P_G is a sequence of probabilities for a sentence. Each element represents the "*K*/*M*" probability for a certain token.

For the discriminator, we still leverage BERT to encode it and use the first token in the last layer to represent the sentence, i.e., $\mathbf{h}_1$. A fully connected layer is employed to help the model identify the style of the sentence:

$$P_D = \text{Classifier}(\mathbf{h}_1) = \text{softmax}(\mathbf{V}\mathbf{h}_1 + \mathbf{b}), \tag{2}$$

where $\mathbf{V}$ and $\mathbf{b}$ are learnable parameters, P_D is the predicted probability of style $\{s_x, s_y\}$. As the traditional adversarial network, the objective of the discriminator is to predict the style of the sentences in corpora correctly:

$$\mathcal{L}_D = -\mathbb{E}_{D,u\sim\{\mathcal{D}_x,\mathcal{D}_y\}}[\log_{P_D}(s_u|u)], \tag{3}$$

where u is the sentence sampled from $\mathcal{D}_x$ or $\mathcal{D}_y$ and s_u is the true label for u. Note that the discriminator needs to be trained in advance, and will be frozen in subsequent training.

3.2.2. Training

Regarding traditional training (Surya et al., 2019) for adversarial generation, given a complex sentence X_i, the output of the generator is the simplified version, denoted as $\hat{Y}_i$. In this case, the discriminator tries to distinguish whether the sentence is simple or not, and the generative network is trained to fool the discriminator, making it difficult to differentiate between simple and complex. However, the above loss is not applicable in our framework due to the following two issues:

1. It is not feasible to include the raw output of the Edit Predictor for adversarial training as "*K*" and "*M*" cannot be directly encoded by the discriminator.

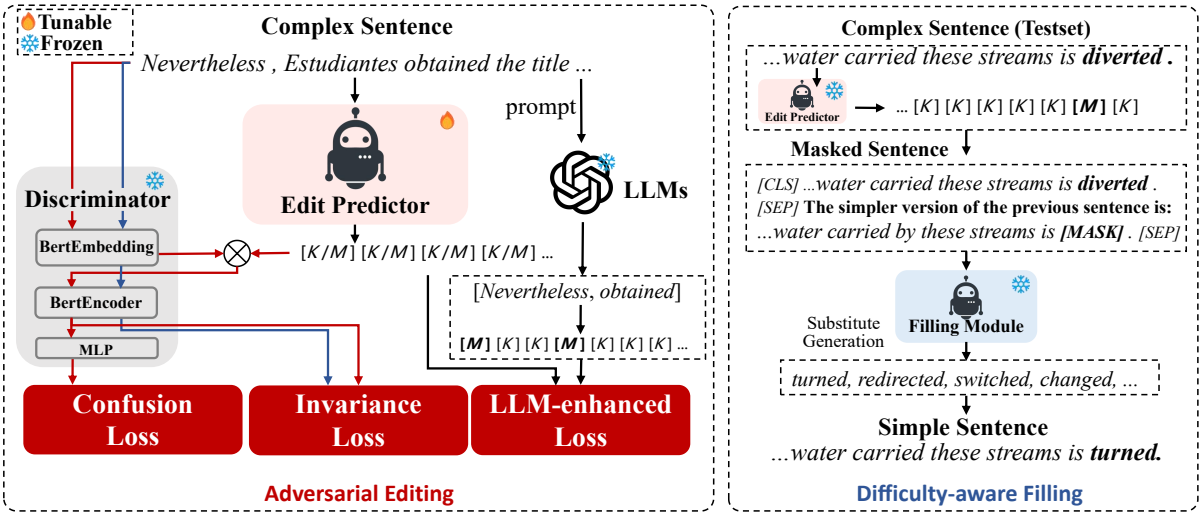

Figure 2: The overall architecture of **LAE-LS**. *Left panel*: **Adversarial Editing** module, where an Edit Predictor predicts lexical edits guided by confusion loss, invariance loss, and LLM-enhanced loss. *Right panel*: **Difficulty-aware Filling** module, in which a Filling Module combines complex sentences and lexical edits into masked sentences as well as generates substitutions for the masked positions, aiming to simplify sentence.

2. It is vital to control the predicted edits and maintain the syntax for lexical simplification. However, existing methods usually ignore this and lead to unexpected changes to the original sentences.

To solve the above issues, we elaborate the objective of Edit Predictor with respect to *confusion loss*, *invariance loss*, and *LLM-enhanced loss*. In other words, the training of the Edit Predictor is guided with the consideration of measuring the confusion degree to the well-performing pre-trained discriminator (*confusion loss*), semantic invariance to the original sentence (*invariance loss*) and similarity to the LLMs' signals (*LLM-enhanced loss*), respectively.

Confusion Loss. To commence, we investigate guiding Edit Predictor in editing complex tokens within a sentence. We aspire to integrate the output of Edit Predictor into the discriminator. To achieve it, we can characterize the sentence representation with the edit labels. Specifically, we multiply the token embedding with the probability of "*K*" predicted by the Edit Predictor, and assume a sentence without a stylistic token will confuse the discriminator by producing a moderate judgement of the transferred sentence.

Formally, the said operation can be represented as:

$$\begin{aligned}\mathbf{w}_l &= \mathbf{w}_l^{tok} \cdot p_l^K + \mathbf{w}_l^{typ} + \mathbf{w}_l^{pos} (l \in [L]),\\ \mathbf{H}^{conf} &= \text{BERT}(\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_L),\\ P_D &= \text{Classifier}(\mathbf{h}_1^{conf}),\end{aligned} \tag{4}$$

where p_l^K is the predicted probability of "*K*" derived from Equation (1), and $\mathbf{h}_1^{conf}$ is the first token embedding. If a token has a low probability of "*K*", the semantic information of the token will be removed from the sentence. For this purpose, we frame a confusion loss $\mathcal{L}_G^{conf}$ is defined below:

$$\mathcal{L}_G^{conf} = (P_D - \alpha)^2. \tag{5}$$

Here, α is a hyper-parameter that we set as an unconfident score. We assume a masked sentence would confuse the discriminator by showing an unconfident score around α instead of $\{0, 1\}$.

Invariance Loss. Regarding the second issue, we incorporate an invariance loss. This additional loss function serves as a reference, encouraging the Edit Predictor to focus on preserving the underlying semantic information of the sentence. The invariance loss, denoted as $\mathcal{L}_G^{inv}$, is computed using the Cosine Similarity metric as outlined below.

$$\mathcal{L}_G^{inv} = 1 - cos(\mathbf{h}_1, \mathbf{h}_1^{conf}). \tag{6}$$

As we can see, when the semantics of the sentences are preserved, the loss tends towards 0, otherwise, it approaches 1.

LLM-enhanced Loss. Large Language Models (LLMs) (Brown et al., 2020), such as ChatGPT or OPT (Zhang et al., 2022), have proven highly effective. A series of approaches manage to leverage LLMs to provide external knowledge for various NLP tasks, such as Question Answering and Information Retrieval (Lan et al., 2023; Guo et al., 2023; Kojima et al., 2022; Wei et al., 2022; Trivedi et al., 2023). As shown in Fig. 2, given the sentence "*Nevertheless , Estudiantes obtained the title at the end of the Apertura 2006 .*", LLMs help to

annotate complex words "*Nevertheless*" and "*obtained*" serving as signals for the Edit Predictor. Motivated by this, we introduce LLM knowledge into the Adversarial Editing System by prompting LLMs, extracting the supervision signals from the responses, and fusing them into the adversarial training procedure.

However, as mentioned above, LLMs are likely to make unexpected edits to the syntax of the sentences. To circumvent over-edit to the original sentences, instead of rewriting the sentences, we prompt LLMs with the instruction as follows:

```
Prompt:
   Please identify the complex words in
↪  the following sentence.
Sentence:
   {Input Sentence}
Output format:
   [w1, w2, ...]
```

In light of the above instruction, LLMs are required to produce complex tokens. Thus, LLM pseudo label loss (termed LLM-enhanced Loss) takes the form:

$$\mathcal{L}_G^{LLM} = -\frac{1}{L}\sum_{w_l \in X_i} [\log_{P_G}(g_l^*|w_l)], \tag{7}$$

where g_l^* is the pseudo gold edit of token w_l generated by LLMs.

Eventually, the overall loss function for the edit predictor is:

$$\mathcal{L}_G = \lambda_1 \mathcal{L}_G^{conf} + \lambda_2 \mathcal{L}_G^{inv} + \lambda_3 \mathcal{L}_G^{LLM}, \tag{8}$$

where λ_1, λ_2 and λ_3 are tunable hyperparameters for balancing different loss items. There is an advantage of distilling the knowledge from LLMs to the Edit Predictor instead of directly leveraging LLMs to make the prediction. That is, LLMs have the risk of over-editing, taking their outputs as the supervision signals play the effect of distilling high-quality knowledge to the small-size models, which can effectively restrain the over-fitting issue (Gou et al., 2021).

3.3. Difficulty-Aware Filling

On top of the well-trained Edit Predictor, which generates a sequence of edit operations, we keep the tokens with "*K*" unchanged and mask the tokens with "*M*". In the example shown in Fig. 2, we obtain "[*Much, of, the, water, carried, by, these, streams, is, M*]" via Edit Predictor. We denote it as $\tilde{X}_i$.

The next procedure is to replace the "*M*" labels with simplified tokens. Existing studies leverage pretrained models to fill in the slots based on the context (Wang et al., 2022; Sun et al., 2023). However, for text simplification, merely including the masked sentences $\tilde{X}_i$ makes the pretrained models ignore the semantic meaning of the complex tokens in the original sentences. Recent effort (Qiang et al., 2021) inputs both X_i and $\tilde{X}_i$ as a pair and feed them into the pre-trained models to predict the tokens at the "*M*" positions in $\tilde{X}_i$. This facilitates pre-trained models to generate tokens with the same semantic meaning as the original complex tokens.

Inspired by this, we introduce a Difficulty-aware Filling module by placing the prompt "*The simpler version of the previous sentence is:* " in between X_i and $\tilde{X}_i$ to encourage the pre-trained model to be aware of the change of the difficulty level. The prompt for the Difficulty-aware Filling module is shown below.

```
Difficulty-aware Filling Prompt:
   [CLS] Original sentence [SEP] The
↪  simpler version of the previous
↪  sentence is:  Masked sentence [SEP]
```

```
Example:
   [CLS] much of the water carried these
↪  streams is diverted . [SEP] The
↪  simpler version of the previous
↪  sentence is: much of the water
↪  carried by these streams is [MASK] .
↪  [SEP]
```

By virtue of the mentioned procedure, we feed the input into the BERT model and let the model predict the words at mask positions. Intuitively, we denote the module as:

$$\hat{Y}_i = \text{Filling\_Module}(\tilde{X}_i). \tag{9}$$

We extract the predicted sentence followed by the instruction as the final simplified sentence $\hat{Y}_i$.

4. Experiments

4.1. Experimental Settings

Datasets. To gauge the effectiveness of our proposed approach, we employ three commonly used datasets for the LS task, namely **LexMTurk** (Colby Horn and Kauchak, 2014), **BenchLS** (Paetzold and Specia, 2016a), and **NNSeval** (Paetzold and Specia, 2016b). These three datasets contain 500, 929, and 239 testing samples, respectively. Also, each sentence in the datasets is annotated with complex words, and multiple alternative simplified words are provided. It is noteworthy that we work on the LS task without parallel corpora thus tap **WikiSmall** (Zhu et al., 2010) as the non-parallel corpus, i.e., $\mathcal{D}_x$ and $\mathcal{D}_y$, which encompasses $84,296$ complex sentences and $84,296$ simple sentences, respectively.

Baselines. We compare our method with a wide range of baselines for CWI, SG, and LS tasks.

1) Evaluation on CWI: Following previous work (Yimam et al., 2017), we use **Character**, **Syllable**, **Vowel**, **Frequency** features and identify complex words via setting a threshold. Moreover, **Attention** (Wang et al., 2022) from a well-trained discriminator can be utilized as it usually assigns more scores to the complex tokens. Additionally, **LSBert** (Qiang et al., 2021) employs BERT as a sequence labeling model and undergoes supervised training on the CWI 2018 dataset (Yimam et al., 2018).

2) Evaluation on SG: A line of methods is given complex words and simply predicts the simplified words. Early methods like **Paetzold-CA** (Paetzold and Specia, 2016), **Paetzold-NE** (Paetzold and Specia, 2017a), and **REC-LS** (Gooding and Kochmar, 2019) utilize word embeddings and rely heavily on parallel corpora or WordNet for assistance. Recently, **LSBert** (Qiang et al., 2021), **BART** (Lewis et al., 2020) and **SimpleBART** (Sun et al., 2023) leverage the mask word prediction capability of pretrained models.

3) Evaluation on LS: Due to the lack of LS methods that integrate CWI and SG, we design several baselines (e.g., **Character-LSBert**, **Syllable-LSBert**, **Vowel-LSBert**, **Frequency-LSBert**, **Attention-LSBert**) for the two stages, building upon **LSBert** (Qiang et al., 2021).

Evaluation Metrics. To fairly compare the performance of different methods, we follow the standard evaluation metrics to measure the CWI and SG (Qiang et al., 2021). Also, we introduce an evaluation metric to evaluate the performance of varying LS systems, which takes two-stage accuracy into consideration. Concretely, we identify a correct prediction when both the complex and top-ranked simplified words are predicted correctly. Hence, we calculate the Precision, Recall and F1 for each sentence and compute the average scores for each test set.

Implementation Details. Unless otherwise stated, we set λ_1, λ_2 and λ_3 all to 1. We fix the unconfident score $\alpha = 0.5$ as default. For Discriminator, we train it with the whole WikiSmall dataset and yield a well-performing Discriminator, which achieves an accuracy of 96.44% on the development set and will be frozen in subsequent training. For Edit Predictor, only complex sentences from the WikiSmall dataset are leveraged. For the Difficulty-aware Filling module, we remove non-English characters and the morphological derivations of the complex words and choose the top 10 words as the substitution following(Qiang et al., 2021). All the models are trained for 30 epochs with batch size of 32. We use Adam optimizer (Kingma and Ba, 2014) with learning rate of $1e$-5. Similar with the setting of prior work (Omelianchuk et al., 2021), we freeze the BERT layer weights during the first four epochs of training, and perform early stopping after 3 epochs account for the performance on the development set. We obtain the SG results of Paetzold-CA, Paetzold-NE, REC-LS, LSBert from (Qiang et al., 2021) and BART, SimpleBART from (Sun et al., 2023), while the remaining models are re-implemented by us.

4.2. Experiment Results

4.2.1. Results Comparison

Comparison with Baselines. We study the performance of different methods for CWI, SG and LS tasks on LexMTurk, BenchLS and NNSeval datasets, as shown in Table 1. The results show that: (1) For the CWI task, which assesses the abilities of models to identify complex words within sentences, LAE-LS achieves the best results on the LexMTurk and NNSeval datasets and demonstrates competitive performance on the BenchLS dataset. One can see that using features (e.g., word character length and syllables) as the simple factors for identifying complex words does not yield satisfactory results. It's worth noting that LSBert is under supervised training, while LAE-LS surpasses it without any annotated parallel corpora. (2) In the SG task, our method outperforms all baselines on the three datasets, thus validating the effectiveness of the Difficulty-aware Filling module. Compared with LSBert, which solely takes a pair of original and masked sentences as the input, LAE-LS exhibits significantly superior performance. This verifies that placing prompts between sentence pairs helps the model be aware of the difficulty for words. (3) Regarding the LS task, our method consistently outperforms the baselines when we integrate CWI and SG together. It is remarkable that there is a performance decrease in terms of F1 for the LS task, compared to CWI and SG tasks. This is because we identify a correct prediction only when complex words are correctly identified and simplified simultaneously, thereby increasing the difficulty of the task. Moreover, we observe that the precision is low for all methods since these methods typically predict more complex words, even though only one is labeled. As such, how to augment the precision in the LS task is challenging.

Comparison with LLMs. Here, we explore the performance of our method and popular LLMs with more parameter size in terms of F1 over LexMTurk dataset. In this end, we select three LLMs with different parameter sizes, including `ChatGLM2` (Du et al., 2021)[1], `llama2` (Touvron et al., 2023)[2],

[1]https://huggingface.co/THUDM/chatglm2-6b

[2]https://huggingface.co/meta-llama/Llama-2-13b-chat-hf

Methods	LexMTurk			BenchLS			NNSeval		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Complex Word Identification									
Character	0.122	0.780	0.211	0.111	0.755	0.194	0.105	0.716	0.183
Syllable	0.140	0.606	0.228	0.117	0.526	0.191	0.100	0.456	0.163
Vowel	0.132	0.764	0.226	0.117	0.727	0.201	0.108	0.678	0.186
Frequency	0.078	0.632	0.139	0.072	0.623	0.129	0.054	0.456	0.096
Attention	0.064	0.512	0.114	0.062	0.448	0.109	0.058	0.435	0.103
LSBert	**0.136**	0.795	0.231	**0.136**	0.788	**0.231**	0.121	0.707	0.207
LAE-LS (ours)	0.135	**0.810**	**0.232**	0.128	**0.813**	0.221	**0.126**	**0.824**	**0.218**
Substitute Generation									
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
LSBert	0.306	0.238	0.268	0.244	0.331	0.281	0.194	0.260	0.222
BART	0.192	0.183	0.188	0.196	0.178	0.192	-	-	-
SimpleBART	0.287	**0.282**	0.285	**0.280**	0.276	0.278	-	-	-
LAE-LS (ours)	**0.340**	0.264	**0.297**	0.262	**0.355**	**0.301**	**0.202**	**0.269**	**0.231**
Lexical Simplification									
Character-LSBert	0.080	0.540	0.139	0.061	0.434	0.107	0.044	0.318	0.078
Syllable-LSBert	0.090	0.410	0.148	0.064	0.299	0.105	0.042	0.201	0.069
Vowel-LSBert	0.087	0.528	0.149	0.063	0.412	0.110	0.045	0.293	0.077
Frequency-LSBert	0.047	0.440	0.085	0.036	0.364	0.066	0.023	0.226	0.042
Attention-LSBert	0.039	0.350	0.070	0.031	0.243	0.054	0.020	0.167	0.036
LSBert	**0.097**	0.564	0.166	0.075	0.454	0.129	0.056	0.335	0.095
LAE-LS (ours)	**0.097**	**0.582**	**0.167**	**0.077**	**0.489**	**0.133**	**0.058**	**0.381**	**0.101**

Table 1: CWI, SG and LS evaluations on three benchmark datasets.

	Size	F1-CWI	F1-SG	F1-LS
ChatGLM2	6B	0.027	0.250	0.048
llama2	13B	0.115	0.264	0.085
GPT-3.5-turbo	175B	0.221	0.296	**0.200**
LAE-LS (ours)	220M	**0.232**	**0.297**	0.167

Table 2: Comparison with various LLMs on LexMTurk Datasets in term of parameter size and F1.

	F1-CWI	F1-SG	F1-LS
LAE-LS (baseline)	**0.232**	**0.297**	**0.167**
w/o LLM-enhanced Loss	0.094	0.297	0.066
w/o Confusion Loss	0.078	0.297	0.135
w/o Invariance Loss	0.089	0.297	0.153
w/o Difficulty-aware Filling	0.232	0.268	0.162

Table 3: Ablation Study of LS on LexMTurk Datasets w.r.t. F1.

`GPT-3.5-turbo` (Brown et al., 2020)[3]. Note that we either employ API or download the checkpoints to do the prediction, and take the responses from the LLMs as the prediction.

[3]https://openai.com/

Specifically, we construct the following prompts for the CWI task:

```
Please identify the complex words in
↪ the following sentence. Sentence:
↪ {Input Sentence}.
```

Similarly, we construct the following prompts for the SG task:

```
Please provide 10 simplified
↪ alternative words for the word
↪ {Complex Word} in the sentence.
↪ Sentence: {Input Sentence}.
```

For the LS task, we regard concatenating the results from the two sub-tasks (including CWI and SG) as it results. The results are displayed in Table 2.

As we can see from the table, LAE-LS, which has a smaller parameter size, can achieve competitive results comparing with the powerful LLMs. For both CWI and SG tasks, LAE-LS trumps all LLMs in terms of F1, indicating that its domination in identifying complex words within sentences and predicting substitutes for each complex word. For the LS task, with respect to F1, LAE-LS leads `ChatGLM2` and `llama2`, but falls slightly behind

Methods	Sentence
Sent (1)	Triangles ... be **classified** according to their internal angles, measured here in degrees.
Candidates	{called, labeled, divided, coded, defined, listed, **categorized**, named, organized, described...}
LSBert	**squares** ... be **categorized** according to their **external triangles**, here in **metric**.
GPT-3.5-turbo	Triangles ... be **categorized** according to their **inner corners**, **calculated** here in **units**.
LAE-LS (ours)	Triangles ... be **categorized** according to their internal angles, measured here in degrees.
Sent (2)	Stone floor tiles tend to ... ceramic tiles and somewhat more **prone** to breakage...
Candidates	{liable, easier, probable, subject, **susceptible**, disposed, **likely**, inclined, vulnerable, apt...}
LSBert	Stone floor tiles tend to ... **porcelain** tiles and somewhat more **susceptible** to **cracking**...
GPT-3.5-turbo	Stone floor tiles tend to ... **clay** tiles and somewhat more prone to **damage**...
LAE-LS (ours)	Stone floor tiles tend to ... ceramic tiles and somewhat more **likely** to breakage...

Table 4: Case study of LS on LexMTurk Datasets. Complex words are highlighted in bold. Candidates indicate the list of annotated simple words for the corresponding complex words in the dataset. Differences between generated and original sentences are in bold.

`GPT-3.5-turbo`. This is because when we compute the F1 score of LS tasks, only the top ranked substitute is taken into consideration while top 10 substitutes are used to measure the F1 score for SG tasks. This indicates that `GPT-3.5-turbo` has the advantage of generating the accurate substitute with top 1 prediction. To sum up, LAE-LS manages to achieve competitive results with a considerably smaller number of parameters compared to `GPT-3.5-turbo`.

4.2.2. Ablation Study

In this section, we systematically delve into the effect of the components in our method by performing the leave-one-out test. Specifically, we iteratively remove the loss functions defined in our Adversarial Editing module. Also, we look into the effectiveness of the Difficulty-aware Filling module. The results are shown in Table 3. From the experimental results, it is evident that removing any of these loss functions leads to performance drop, suggesting that they are vital for the training of Edit Predictor. As we have introduced in Section 3.2.2, they measure the prediction from different aspects. Of note, LLM-enhanced loss plays the most crucial role for training of Edit Predictor. This means that it is effective to distill the knowledge from LLMs to models with smaller size, especially in the absence of annotated data. Moreover, we replace the Difficulty-aware Filling module with LSBert and the results show there is a notable decrease on the F1 score of the SG task. The above results indicate that our proposed Difficulty-aware Filling module indeed guides the pretrained model to generate accurate words rather than words that do not preserve the original meaning or not simple enough.

4.2.3. Case Study

To further go into the virtue of LAE-LS, we present multiple case studies in Table 4. For sentence (1), we can observe that all models successfully identify "*classified*" as a complex word and replace it correctly with "*categorized*". However, in the case of LSBert, it identifies "*triangles*" as a complex word, which is inaccurate because "*triangles*" is a non-stylistic word that holds the semantic information of the sentence. We suspect that LSBert is trained on supervised CWI data, focusing on the inherent difficulty of words and ignoring the semantic meaning of them in the sentence. In contrast, our Edit Predictor is measured by the invariance loss, which preserves the semantic information of the original sentence effectively. Notably, compared with `GPT-3.5-turbo`, which tends to identify more complex words and oversimplify the sentence, LAE-LS incorporates adversarial training, which enables to more accurate complex words identification and prevents excessive modifications in the sentence. For sentence (2), apart from `GPT-3.5-turbo`, both LAE-LS and LSBert accurately identify this complex word, though LSBert tends to label many other words as complex words. Significantly, LSBert simplifies *"prone"* to "susceptible," while LAE-LS simplifies it with a much simpler word *"likely"*. This suggests that our method considers not only the semantic meaning of the context but also the complexity of the generated words, thus leading to a more desirable prediction.

5. Conclusion

In this paper, we propose an LLM-enhanced Adversarial Editing System to address the lexical simplification task without parallel corpora, which consists of an Adversarial Editing module and a

Difficulty-aware Filling module. Adversarial Editing module is guided by a confusion loss and an invariance loss to make lexical edits with a consideration of semantic preservation and simplified ratio. Meanwhile, we craft an LLM-enhanced loss to distill knowledge from LLMs, thus further augmenting the Adversarial Editing module. From that, the Difficulty-aware Filling module combines the original sentences and lexical edits to mask complex words within sentences and fill in the masked positions with simpler words. The extensive experimental results on three LS datasets demonstrate that our method is effective. That is, our method not only advances lexical simplification in the absence of parallel corpora but also showcases the potential for leveraging the capabilities of large language models to enhance the simplification process.

Ethics Statement

This is a study about Lexical Simplification. It does not have any data privacy issues. We did not collect any personal information. This is a task that involved no risk as the participants were not exposed to any harmful material or asked to perform any risky tasks.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments. This work was supported by Joint Key Project (Project No. U23A20298) and Young Scientists Project (Project No. 62206097) of National Natural Science Foundation of China, Shanghai Pujiang Talent Program (Project No. 22PJ1403000) and East China Normal University (Project No. 2022ECNU—WHCCYJ-31).

Bibliographical References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Alison Chi, Li-Kuang Chen, Yi-Chen Chang, Shu-Hui Lee, and Jason S Chang. 2023. Learning to paraphrase sentences to different complexity levels. *arXiv preprint arXiv:2308.02226*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Siobhan Devlin. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Yutao Feng, Jipeng Qiang, Yun Li, Yunhao Yuan, and Yi Zhu. 2023. Sentence simplification via large language models. *arXiv preprint arXiv:2302.11957*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Sian Gooding and Ekaterina Kochmar. 2018. Camb at cwi shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.

Sian Gooding and Ekaterina Kochmar. 2019. Recursive context-aware lexical simplification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4853–4863.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819.

Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven Hoi. 2023. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10867–10877.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting proper lexical paraphrase for children. In *Proceedings of the 25th Conference on Computational*

Linguistics and Speech Processing (ROCLING 2013), pages 59–73.

Robin Keskisärkkä and Arne Jönsson. 2012. Automatic text simplification via synonym replacement. *SLTC 2012*, page 47.

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A Hearst. 2021. Keep it simple: Unsupervised simplification of multi-paragraph text. *arXiv preprint arXiv:2107.03444*.

Yunshi Lan, Xiang Li, Xin Liu, Yang Li, Wei Qin, and Weining Qian. 2023. Improving zero-shot visual question answering via large language models with reasoning question prompts. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 4389–4400.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Yuanyuan Liang, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. 2023. Prompting large language models with chain-of-thought for few-shot knowledge base question generation. *arXiv preprint arXiv:2310.08395*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Kostiantyn Omelianchuk, Vipul Raheja, and Oleksandr Skurzhanskyi. 2021. Text simplification by tagging. *arXiv preprint arXiv:2103.05070*.

Gustavo Paetzold and Lucia Specia. 2017a. Lexical simplification with neural ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 34–40.

Gustavo H Paetzold and Lucia Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3761–3767.

Gustavo H Paetzold and Lucia Specia. 2017b. A survey on lexical simplification. *Journal of Artificial Intelligence Research*, 60:549–593.

Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, Yang Shi, and Xindong Wu. 2021. Lsbert: Lexical simplification based on bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3064–3076.

Jipeng Qiang, Kang Liu, Yun Li, Yunhao Yuan, and Yi Zhu. 2023. ParaLS: Lexical substitution via pretrained paraphraser. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3731–3746. Association for Computational Linguistics.

Sandaru Seneviratne, Elena Daskalaki, and Hanna Suominen. 2022. Cils at tsar-2022 shared task: Investigating the applicability of lexical substitution methods for lexical simplification. In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 207–212.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, 30.

Renliang Sun, Wei Xu, and Xiaojun Wan. 2023. Teaching the pre-trained model to generate simple texts for text simplification. *arXiv preprint arXiv:2305.12463*.

Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. 2019. Unsupervised neural text simplification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2058–2068. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay

Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037. Association for Computational Linguistics.

Jiarui Wang, Richong Zhang, Junfan Chen, Jaein Kim, and Yongyi Mao. 2022. Text style transferring via adversarial masking and styled filling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7654–7663.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Rodrigo Wilkens, David Alfter, Rémi Cardon, Isabelle Gribomont, Adrien Bibal, Watrin Patrick, Marie-Catherine de Marneffe, and Thomas François. 2022. Cental at tsar-2022 shared task: How does context impact bert-generated substitutions for lexical simplification? In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 231–238.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - complex word identification task across three text genres and two user groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407. Asian Federation of Natural Language Processing.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9668–9675.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361.

Language Resource References

Colby Horn, Cathryn Manduca and David Kauchak. 2014. *Learning a lexical simplifier using Wikipedia*. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016a. *Benchmarking Lexical Simplification Systems*. European Language Resources Association (ELRA).

Gustavo Paetzold and Lucia Specia. 2016b. *Unsupervised lexical simplification for non-native speakers*. AAAI Conference on Artificial Intelligence.