Divide-and-Conquer Text Simplification by Scalable Data Enhancement

Sanqiang Zhao
Amazon Alexa AI
sanqiang@amazon.com*

Rui Meng
Salesforce
ruimeng@salesforce.com†

Daqing He
University of Pittsburgh
daqing@pitt.edu

Hui Su
Wechat AI
aaronsu@tencent.com

Abstract

Text simplification, whose aim is to reduce reading difficulty, can be decomposed into four discrete rewriting operations: substitution, deletion, reordering, and splitting. However, due to a large distribution discrepancy between existing training data and human-annotated data, models may learn improper operations, thus lead to poor generalization capabilities. In order to bridge this gap, we propose a novel data enhancement method, SimSim, that generates training pairs by simulating specific simplification operations. Experiments show that the models trained with SimSim outperform multiple strong baselines and achieve the better SARI on the Turk and ASSET datasets. The newly constructed dataset SimSim is available at https://github.com/Sanqiang/sent_simplicaition_data.

1 Introduction & Related Work

Text simplification is a task to reduce the complexity of a text while retain its original meaning. It can facilitate people with low-literacy skills or language impairments, such as children and individuals with dyslexia (Rello et al., 2013) and aphasia (Carroll et al., 1999), to read and understand complicated materials (Watanabe et al., 2009). Normally, substitution, deletion, reordering, and splitting are considered as four core operations for performing text simplification (Zhu et al., 2010). Thus an ideal model should be capable of executing these operations appropriately to simplify a text. However, by examining the degree that each operation is exerted in different datasets, we observe that there is an salient discrepancy between the human annotation and existing training data that is widely used for training simplification models. To alleviate this discrepancy, we propose an unsupervised data construction method that distills each simplifying operation into data via different automatic data enhancement measures. The empirical results demonstrate that the resulting dataset SimSim can support models to achieve better performance by performing all operations properly.

2 Inspecting Simplification Datasets

At its essence, sentence simplification paraphrases a sentence for better readability. It often involves a subset of four rewriting operations/transformations: **splitting**, **dropping**, **reordering**, and **substitution** (Zhu et al., 2010; Zhang and Lapata, 2017). A high-quality sentence simplfication training dataset, which contains many complex-simple sentence pairs, should be well-aligned to provide a wide coverage of different operations, so that the trained models can have good generalizability. Most neural simplification models rely on training with large datasets such as Newela (Xu et al., 2015) and WikiLarge (Zhang and Lapata, 2017), which are automatically collated with paired documents written in different readability levels. The quality of auto-collated data has been questioned in prior work (Jiang et al., 2020), however, it remains unanswered on how well they represent the real simplification distribution and on which aspects they fall short. This motivated us to propose the following five metrics to quantitatively examine common training datasets:

2.1 Measuring Simplification Operations

Alignment between the pair of complex/simplified sentence is a fundamental property since the latter should preserve the meaning of the former. Most datasets are collated from paired complex-simple documents using automatic alignment algorithms (Zhu et al., 2010; Xu et al., 2015), their sentence pairs can be poorly aligned. This is because editors may restructure the words, sentences, or even paragraphs drastically when rewriting a text into different readability level. In consequence, sentences in a paraphrased

[*] work was done at University of Pittsburgh
[†] work was done at University of Pittsburgh

Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022), pages 166 - 172
December 8, 2022 ©2022 Association for Computational Linguistics

document may not accurately pair with the original ones. We adopt BERTScore (Zhang* et al., 2020) to measure the semantic alignment between a complex and a simple sentence.

Substitution denotes replacing complicated words or phrases with simplified synonyms. We adopt PPDB (Pavlick et al., 2015) to measure the amount of substitutions between two sentences. PPDB provides extensive substitution rules (see examples in Table 1) and we measure the degree of substitution by checking the ratio of simplified tokens in a sentence pair (normalized by the length of s_{simp})).

Weight	Type	Rule
0.99623	[VP]	recipient → have receive
0.75530	[NN]	recipient → winner
0.58694	[NN]	recipient → receiver

Table 1: Example of simplifying rules in PPDB

Dropping refers to the rewriting transformation by removing unimportant or redundant parts from a sentence. To measure the degree of dropping, we calculate the ratio of the tokens being discarded from a complex sentence.

Reordering denotes the rearrangement of parts in a sentence to simplify its syntax and structure. To measure the reordering, we extract the syntactic structure of each sentence and compare the syntactical change between each pair of sentences.

Concretely, following the method proposed by Xu et al., we use a dependency parser (Honnibal et al., 2020) to extract dependency relations from a sentence. Then Jaccard similarity between the two sets of relations is calculated to measure the degree of reordering transformation.

Splitting divides a long sentence into several shorter ones to reduce syntactic complexity. We count the number of sentences on both sides, and a help from the split is observed when the number of sentences at the simplified side is larger.

2.2 Studies on Existing Datasets

We conducted quantitatively inspections using the five proposed metrics on four mainstream datasets: WikiLarge and Newela, which are commonly used as training data in prior work, as well as the validation set of Turk (Xu et al., 2016) and ASSET (Alva-Manchego et al., 2020). The latter two were annotated by human and are representative of real distribution of text simplification. Figure 1 shows the results on four metrics. On *alignment*, Turk and ASSET contain most aligned sentence pairs and almost all sentence pairs are of high similarity (larger than 0.9), whereas WikiLarge and Newela have a large proportion of poorly aligned pairs with WikiLarge is more problematic. Turk and ASSET also present more *substitution* than the two other datasets, and WikiLarge exhibits a very different distribution of *dropping* from the others, where it discards more words and in certain extreme cases only retains a few words at the simplified side. Lastly, Turk and ASSET contain less *reordering* (leaning towards 1.0), whereas WikiLarge and Newela contain sentences with drastic syntactic changes. Table 2 shows the proportion of sentence pairs contribute to splitting. A large proportion of sentence pairs in ASSET help to split, which indicates the splitting cannot be ignored but all other datasets rarely help to split.

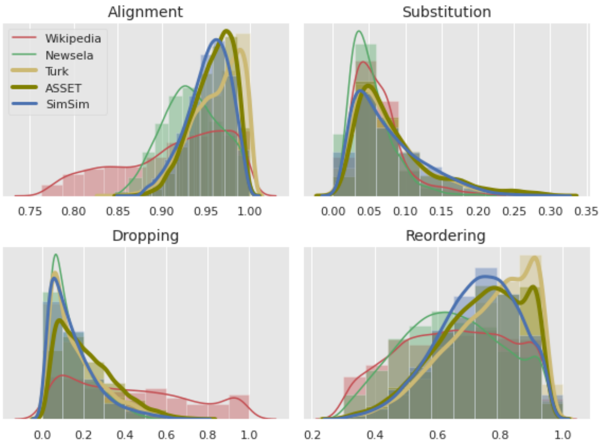

Figure 1: The histograms and density estimates of four property measures in simplification data.

Corpus	Proportion
Wikipedia	0.102
Newsela	0.002
Turk	0.044
ASSET	0.310
SimSim	0.399

Table 2: Proportion of sentence pairs helps to split

Overall, a large discrepancy in the rewriting distributions is shown between Turk, ASSET and the two training datasets, which makes us wonder the validity of the models trained with such biased data. This motivates us to develop novel training data that can better transfer real knowledge of text simplification to models.

3 SimSim: Data Enhancement by Simulating Simplification

We think that a well-generalizable simplification model needs to be trained on high quality training pairs and simplification knowledge. we propose a method to automatically refine/construct existing training pairs and inject knowledge of simplification by simulating various rewriting transformations. Unlike previous datasets (i.e. WikiLarge and Newela) that heavily rely on paired documents of different readabilities and can hardly scale up, our method is capable of exploiting any text data on the Internet as seed thus avoid those limitations. We name the resulting dataset SimSim.

Overview Our method starts with a set of seed sentences, then a series of enhancement steps are performed on each seed sentence to generate a new sentence so that the original seed sentence and the new resulting sentence form a complex-simple pair. Note that this method not only can enhance original training pairs, it can also construct new pairs solely using complex sentences. This method works on other corpora too, but we leave it for future work. To build seed sentences, we apply BERTScore on each sentence pair in WikiLarge and Newela to check their semantic alignment between the complex and simple sentences. For those badly aligned pairs, we remove their simple sentences to eliminate the noise led by the misalignment. We take the rest sentences as seeds for enhancement.

Constructing Paraphrastic Sentences by Back-Translation The bottom line of simplifying a sentence is to paraphrase it without alternating its meaning. Rather than retrieving aligned sentences from paired documents, we propose to create such pairs with the help of back-translation. We expect that the back-translated sentences should preserve the meaning of the original sentences, meanwhile demonstrate more linguistic diversity. The idea has been proven effective for paraphrasing sentences (Wieting et al., 2017) and improving translation with monolingual data (Sennrich et al., 2016).

We employ Google's Neural Machine Translation System (GNMT) (Wu et al., 2016) for this purpose, on account of its overall translation quality and the support of a large number of languages. We translated each seed sentence into 103 pivot languages and translated it back to English. Some examples are shown in Table 3.

Candidate Selection with GPT-2. Paraphrastic sentences generated through back-translation can contain language errors and unnatural expressions. GPT-2, as a powerful neural language model trained with open-domain text (Radford et al., 2019), can help us to evaluate the quality of candidate sentences. GPT-2 gives a score (negative log-likelihood) to a sentence, and we assume that a better GPT-2 score means that the sentence is more likely to be in high quality. Thus among all 103 candidates, we select the best one as the target sentence for further enhancement. Particularly, if GPT-2 deems the back-translated sentence less natural than the original one, it will be discarded. Although, the remaining candidate sentences after GPT-2 scoring can be considered to be well-aligned and natural, they are not ready for training a simplification model since most of them have not been simplified yet. They therefore go through a set of simulating steps as presented below:

Simulating Substitution In order to impose substitution knowledge into the candidate sentences, we applied the paraphrasing rules in PPDB. To ensure the applied rules are proper, we use GPT-2 again to evaluate the quality of the resulting sentences.

Simulating Dropping To distill the dropping operation into data, we follow previous approach (Filippova and Altun, 2013) and augment the data by randomly removing prepositional, adjective or adverb phrases.

Simulating Splitting We find that back-translation rarely splits a sentence into multiple shorter ones. Thus we propose to include WikiSplit (Botha et al., 2018) to incorporate the splitting operation into our data. We put WikiSplit sentence pairs into the seed bank and we apply the above process to the target-side sentences so as to mix the splitting transformation with others.

SimSim Dataset By simulating different operations with the above steps, we present a new corpus SimSim for the task of text simplification. As shown in Figure 1 and Table 2, SimSim demonstrates a closer distribution to the human-annotated Turk and ASSET, from multiple rewriting aspects. This suggests that SimSim may serve as a better dataset for training simplification models.

4 Experiments

Setup We train Transformer-based vanilla Encoder-Decoder models with five datasets. The

Pivot	Sentence	NLL
Original	It is situated at the coast of the Baltic sea , where it encloses the city of stralsund .	3.8020
Chinese	It is located on the coast of the Baltic Sea and surrounds the city of Stralsund .	2.8642
Greek	It is located on the shores of the Baltic Sea, where it encloses the city of Stralsund .	2.8379
Italy	It is located on the Baltic Sea coast, where the city of Stralsund is located .	2.9493
Japanese	It is located on the Baltic Sea coast and surrounds the city of Stralsund .	3.1864
Hindi	It is situated on the banks of the Baltic sea, where it surrounds the town of Stralsund .	3.0487

Table 3: Examples of back-translation with GNMT. The rightmost column shows the Negative log-likelihood (NLL) scores estimated by GPT-2.

first two are WikiLarge and Wiki-Auto, two common training datasets. **WikiLarge** (Zhang and Lapata, 2017) is constructed by automatically aligning sentences in Simple Wikipedia and Wikipedia, with the help of lexical-based features, such as the Jaccard coefficient and TF-IDF. It has 296k complex-simple sentence pairs. **Wiki-Auto** (Jiang et al., 2020) uses a neural CRF model in order to achieve a better auto-alignment than the rule-based method used in WikiLarge, which contains 488k pairs. The remaining three datasets are the three variants of SimSim dataset: (1) SimSim-S1, constructed by directly applying 103-language back-translation on candidate sentences, resulting millions of pairs; (2) SimSim-S2, constructed by selecting the most natural sentence from translated sentences with GPT-2 (1.67M pairs); (3) SimSim-S3 further improves SimSim-S2 by simulating different rewriting operations (1.67M pairs). We use Turk (Xu et al., 2016) and ASSET (Alva-Manchego et al., 2020) for validation and testing. SARI (Xu et al., 2016) is used as the evaluation measure since it is widely used in the literature.

Train Data	SARI↑			
	Score	Add	Delete	Keep
SBMT-SARI (Xu et al., 2016)	39.96	5.96	41.42	72.52
DMASS-DCSS (Zhao et al., 2018)	40.45	5.72	42.23	73.41
EditNTS (Dong et al., 2019)	38.23	3.36	39.15	72.13
Edit-Unsup-TS (Kumar et al., 2020)	37.85	2.31	43.65	67.59
WikiLarge	38.84	4.78	41.19	70.53
Wiki-Auto	39.64	5.18	41.61	72.13
SimSim-S1	36.33	4.53	32.79	71.66
SimSim-S2	40.15	7.52	38.64	**74.32**
SimSim-S3	**41.07**	**8.33**	**41.97**	72.89

Table 4: Performance of vanilla Encoder-Decoder models and some other baselines tested on WikiTurk dataset.

Train Data	SARI↑			
	Score	Add	Delete	Keep
Wiki-Auto	50.79	16.65	69.58	66.16
SimSim-S1	49.34	17.10	66.48	64.44
SimSim-S2	52.20	**19.34**	69.89	**67.38**
SimSim-S3	**52.37**	19.18	**71.01**	66.92

Table 5: Performance of vanilla Encoder-Decoder models tested on ASSET dataset.

Results The experiment results are presented in Table 4 and 5. Between two models trained with Wiki-Auto and WikiLarge respectively, the one on Wiki-Auto achieved better scores than that of WikiLarge, which is helped by the improved aligning algorithm. However, Wiki-Auto is still limited to the contents in Wikipedia and contains much noise. In comparison, models trained with SimSim outperform WikiLarge and Wiki-Auto consistently. Because SimSim is constructed in a more controlled way, the sentences in each pair are more aligned and more rewriting operations are included. Among the three SimSim variants, SimSim-S1, constructed by only back-translation, performs the worst among the three, and worse than two baselines. Back-translation itself can boost the diversity of sentence pairs, nevertheless it also introduces much noise in language. By utilizing GPT-2 to select the most natural ones from back-translated pairs, the model using SimSim-S2 outperforms SimSim-S1 by a large margin. Moreover, the SARI performance can be further boosted with SimSim-S3, which applied multiple rewriting operations on each training pair to simulate the real simplification process.

5 Conclusion

In this study, we observe a significant discrepancy exists between the human simplified sentences and common training data and propose an unsupervised data enhancement method, SimSim, to explicitly teach the model appropriate operations by distilling the knowledge into training data. The empirical results show that the resulting dataset SimSim can support models to achieve better performance by performing all operations properly.

References

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679.

Jan A Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. 2018. Learning to split and rephrase from wikipedia edit history. *arXiv preprint arXiv:1808.09468*.

John A Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *EACL*, pages 269–270.

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. *arXiv preprint arXiv:1906.08104*.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural crf model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7943–7960.

Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative edit-based unsupervised sentence simplification. *arXiv preprint arXiv:2006.09639*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 425–430.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *URL https://openai. com/blog/better-language-models*.

Luz Rello, Clara Bayarri, Azuki Gòrriz, Ricardo Baeza-Yates, Saurabh Gupta, Gaurang Kanvinde, Horacio Saggion, Stefan Bott, Roberto Carlini, and Vasile Topac. 2013. Dyswebxia 2.0!: more accessible text for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 25. ACM.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 274–285.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594.

Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. 2018. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*.

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*, pages 1353–1361. Association for Computational Linguistics.

A Training Details

Our Transformer architecture uses an embedding dimension of 512, fully connected layers of dimension 2048, 8 attention heads, 6 layers in the encoder and 6 layers in the decoder. we used beam search with a beam size of 8. For optimization, model updates use a batch size of 400 and a LAMB optimizer with learning rate 0.001 (You et al., 2019) and all the models were trained by 250,000 steps (takes around 2-3 days). Training was done on a single Cloud TPU V2.

B Evaluation Details

We computed SARI using the code provided at `https://github.com/cocoxu/simplification` and we mainly compare our results with studies using the same evaluation protocal (Xu et al., 2016; Zhang and Lapata, 2017; Zhao et al., 2018; Dong et al., 2019; Kumar et al., 2020). Note that most studies reported ASSET scores using a different evaluation code and therefore we cannot include their scores for the sake of fair comparison.

C Implementation Details

Our model implementation is based on Tensorflow 1.15 and Tensor2Tensor library `https://github.com/tensorflow/tensor2tensor`.