

# Can Knowledge Graphs Simplify Text?

Anthony Colas\*  
acolas1@ufl.edu  
The University of Florida  
Gainesville, Florida, USA

Haodi Ma\*  
ma.haodi@ufl.edu  
The University of Florida  
Gainesville, Florida, USA

Xuanli He  
xuanli.he@ucl.ac.uk  
University College London  
London, United Kingdom

Yang Bai  
baiyang94@ufl.edu  
The University of Florida  
Gainesville, Florida, USA

Daisy Zhe Wang  
daisyw@cise.ufl.edu  
The University of Florida  
Gainesville, Florida, USA

## ABSTRACT

Knowledge Graph (KG)-to-Text Generation has seen recent improvements in generating fluent and informative sentences which describe a given KG. As KGs are widespread across multiple domains and contain important entity-relation information, and as text simplification aims to reduce the complexity of a text while preserving the meaning of the original text, we propose **KGSimple**, a novel approach to unsupervised text simplification which infuses KG-established techniques in order to construct a simplified KG path and generate a concise text which preserves the original input’s meaning. Through an iterative and sampling KG-first approach, our model is capable of simplifying text when starting from a KG by learning to keep important information while harnessing KG-to-text generation to output fluent and descriptive sentences. We evaluate various settings of the **KGSimple** model on currently-available KG-to-text datasets, demonstrating its effectiveness compared to unsupervised text simplification models which start with a given complex text. Our code is available on [GitHub](#)<sup>1</sup>.

## CCS CONCEPTS

• **Computing methodologies** → **Natural language generation; Knowledge representation and reasoning**; • **Theory of computation** → *Graph algorithms analysis*.

## KEYWORDS

Knowledge Graph; Data-to-Text; Natural Language Generation; Text Simplification; KG-to-Text; Simulated Annealing

## ACM Reference Format:

Anthony Colas\*, Haodi Ma\*, Xuanli He, Yang Bai, and Daisy Zhe Wang. 2023. Can Knowledge Graphs Simplify Text?. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, October 21–25, 2023, Birmingham, United Kingdom. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3583780.3615514>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or to publish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*CIKM '23, October 21–25, 2023, Birmingham, United Kingdom*  
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 979-8-4007-0124-5/23/10...\$15.00  
<https://doi.org/10.1145/3583780.3615514>

## 1 INTRODUCTION

Text simplification (TS) is characterized as a text revision task, with the constraint that the output text should be easier to read than the input text. The primary objective of TS is to propagate information to a more expansive audience, including individuals with lower literacy levels [47], those with reading disabilities [7], non-native speakers [37], and individuals lacking specialized knowledge within specific domains, such as medically related documents [1, 52]. Therefore it can also enhance various natural language processing (NLP) tasks that necessitate less complex texts for optimal results, including question answering [10], information extraction [51], and machine translation [48].

Models employing generative supervised learning for TS have typically adhered to a sequence-to-sequence framework, whereby a complex body of the text is translated into a simplified sentence in an autoregressive fashion [37, 59]. Conversely, unsupervised models typically depend on parsing and restructuring sentences to generate text that is both simpler and semantically relevant [12, 26]. The generative models frequently surpass their counterparts that depend on editing mechanisms and rules for sentence simplification, although they necessitate the availability of abundant parallel sentence pairs for effective training, which may not be consistently accessible. Moreover, while supervised models execute the translation of complex sentences through a series of continuous steps, unsupervised TS has predominantly concentrated on employing edit-based approaches. These approaches involve rule-driven operations, including sentence reordering, splitting, and deletion, providing a higher level of interpretability, but are inherently limited by their predefined rule sets.

Recent investigations into unsupervised text simplification (TS) have delved into the domain of search-based methodologies. These methods utilize predefined scoring functions that prioritize factors such as simplicity, fluency, and meaning retention in a sentence as per the studies conducted by Dehghan et al. [11], Laban et al. [27], Vo et al. [56]. Similar to supervised counterparts, these methodologies can also incorporate a generation phase to maintain the integrity of sentence structure. Despite these advancements, current unsupervised strategies are limited in their capacity to generate highly fluent sentences. This limitation arises due to the constraints

\* These authors contributed equally.

<sup>1</sup><https://github.com/acolas1/KGSimple>

imposed by edit operations on the structural and syntactical elements of the input sentences.

As there is a vast amount of data stored in knowledge graphs (KGs), including patient records [45], legal procedures [14], and event-related information [9] it is important to convey such information to a wide audience. KG-to-text models have seen recent improvements in transcribing such structured data into fluent natural language sentences which preserve the meaning of the KG [8, 21]. Still, because of the vast amounts of data stored in KGs, some of the data may be irrelevant and difficult to understand when transcribing to a user. Furthermore, as the data stored in KGs can act as a compact representation of narratives, the data stored in KGs can be used to help interpret such TS models, especially those that leverage a text generation step. Thus, we propose an unsupervised framework that infuses TS with KG-to-text generation to generate simplified texts that preserve the meaning of the original input KG while generating more fluent texts than existing unsupervised TS approaches. Whereas current TS approaches rely on input texts, we extend TS to the KG domain. Our framework, which we call KGSimple, is a sampling-based approach that utilizes graph operations to simplify the input’s contents while leveraging pre-trained KG-to-text generative models to produce fluent and relevant text with respect to the input KG. An example is illustrated in Figure 1. We adopt an iterative graph sampling to produce simplified KGs akin to Markov Chain Monte Carlo (MCMC) sampling, where at each step a local operation is performed on a KG to reduce the structural and syntactic complexity of the graph. We then generate a text from the proposed graph, either accepting or rejecting the current KG-text pair according to a heuristically defined scoring function. We explore multiple sampling algorithms to produce a simplified text, including simulated annealing (SA) [53] to search for a simplified KG.

Our contributions are as follows:

- We propose KGSimple, an unsupervised KG-text simplification framework that can produce simplified text based on an input KG. Note, that our algorithm can produce both sentence and paragraph-level simplification.
- We devise up to three sampling techniques to simply more complex KGs and integrate existing KG-to-text models on the simplified KGs to generate coherent and simplified texts.
- We experiment on existing KG-to-text data and compare it to existing text-centric unsupervised TS systems, demonstrating that KGSimple can at times outperform these models, specifically in generating fluent simplified text in an interpretable fashion.

## 2 RELATED WORK

### 2.1 Text Simplification

A variety of generative supervised models have been employed, including sequence-to-sequence models [37], reinforcement learning techniques [62], and transformer architectures [54]. These approaches have utilized external paraphrase databases [63], complexity-weighted loss [25], syntactic rules [31], and complexity features found within the text [32] to improve their performance.

On the other hand, edit-based supervised models have been developed to simplify complex sentences by leveraging parallel

complex-simple sentences. Alva-Manchego et al. [3] have proposed a method that learns the keep, replace, and delete operations, while Dong et al. [12] have developed an end-to-end generative model to learn where to apply edit operations. Recent work on supervised edit-based text simplification has focused on predicting token-level edit-operations in a non-autoregressive fashion [39] or editing the complex text through a fixed pipeline [2]. However, unlike KGSimple, these approaches require large amounts of parallel supervised data and focus mainly on sentence and word-level edits. In contrast, our KG-level edits can produce changes in the output at various granularities.

Several semi-supervised methods have been proposed for text simplification. For example, Zhao et al. [64] introduced a back-translation framework, while Surya et al. [49] employed a style-transfer technique, and Martin et al. [33] fine-tuned BART approach. However, these methods still require non-aligned pairs of sentences, which limits their interpretability and controllability since they are based on generative networks. On the other hand, unsupervised edit-based approaches have been explored, where a pipeline of operations is applied to complex sentences [36].

Recently, iterative approaches have been proposed, where text simplification is modeled as a search problem [26]. These methods have integrated text generation [11, 27] and back-translation [56] as a step in their framework. Although KGSimple is also an iterative revision-based approach, it differs from these methods in that it leverages the knowledge graph’s condensed storage of information as a starting point instead of plain text. Moreover, while previous work on text simplification has mostly focused on greedy selection processes, KGSimple considers various sampling strategies, inspired by adjacent text generation tasks[30].

### 2.2 KG-Text Generation

Previous research on models that convert KG into natural language text first employed GNNs to encode the neighborhood structure of the graph before decoding the information into text [17, 24, 55]. Recent studies in the field of KG-to-text have explored the efficacy of pre-trained language models [44] and have developed pre-training tasks to acquire knowledge of the KG representation [19]. These models have also incorporated graph-based features into transformers, such as a node’s relative position [46] and the graph’s topology [8, 21], in order to enhance KG encoding. While the previously mentioned research has concentrated on narrating the KG, we are the first to apply such KG-to-text models to unsupervised text generation, specifically, text simplification.

## 3 APPROACH

### 3.1 Overview

The proposed framework, KGSimple, comprises a two-fold approach that employs an iterative process to optimize a given reward function. As depicted in Figure 2, given an input KG  $g$ , the framework performs the following steps: (1) sample a simplified graph  $g'$  through diverse graph reduction operations on the structure and syntactic content of the original graph (refer to Section 3.2), and (2) generate a corresponding text  $y_{g'}$  via a KG-to-text generator (refer to Section 3.3). Both  $g$  and  $y_{g'}$  are then evaluated based on a pre-defined scoring function (refer to Section 3.4).

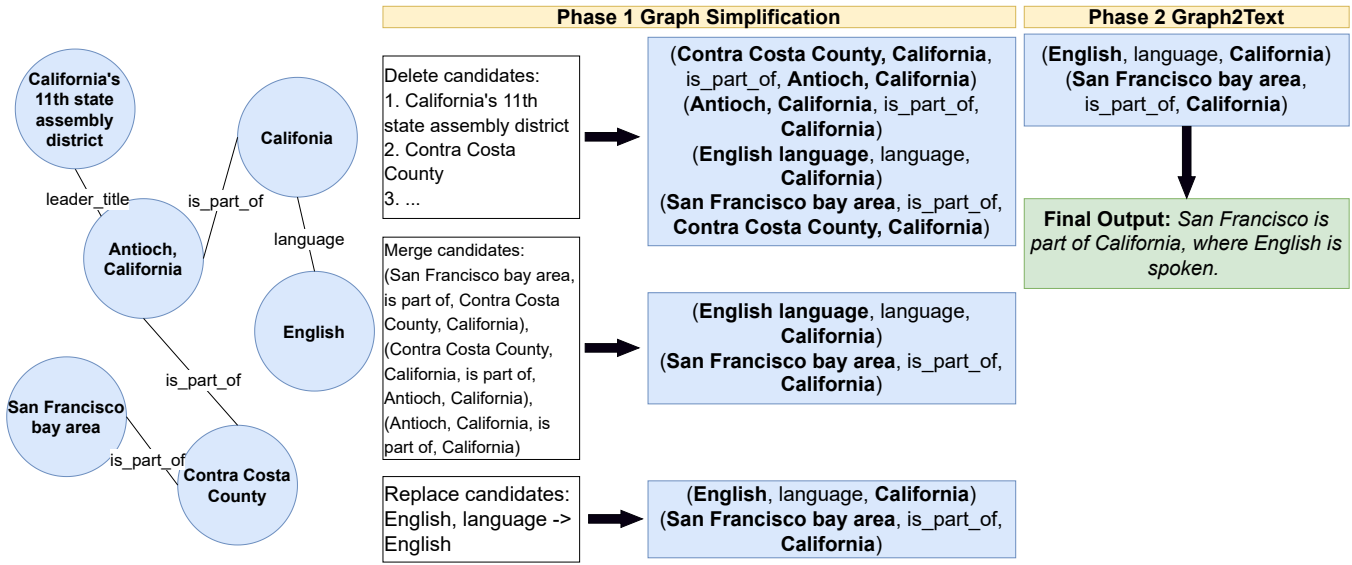


Figure 1: Illustration of the KGSimple framework: from input KG to generated text.

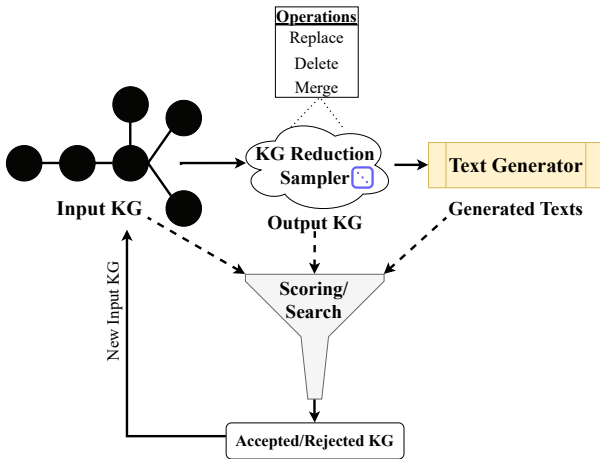


Figure 2: The KGSimple framework: 1) the KG Reduction Sampler produces a simplified KG, 2) the text generator (GAP) produces a text representation, and 3) the scoring function evaluates the KG-text pair according to a given condition.

At each iteration of the framework, starting with KG  $g_0$ , a reduction operation is applied to the KG, and a pre-trained generative model produces a text that corresponds to the current KG. Our scoring function considers both soft and hard constraints on the KG-Text pair and determines whether to retain or discard the proposed KG-Text pair for use in the next iteration. If the KG-Text pair is rejected, KGSimple will randomly sample another operation on the KG to generate a new text from the generator.

Our scoring function is heuristically defined, and therefore, to ensure the credibility of our results, we employ three distinct acceptance criteria outlined in Section 3.4. First, we accept scores that are greater than or equal to zero. Second, we consider

scores greater than the previous score to indicate progress. Finally, we utilize simulated annealing (SA) as a technique to encourage the model to explore beyond local minima. By employing these acceptance criteria, we aim to ensure the robustness and accuracy of our scoring system.

### 3.2 Operations

We detail the operation set sampled by KGSimple that simplifies a KG at each step. We propose candidates by sampling one of the operations that can modify entities/relation tokens and reduce the structure of the original KG.

**Delete.** In this operation, we propose a new candidate KG by eliminating one of the entities in the original KG. To ensure the preservation of crucial information after each deletion, we utilize a TF-IDF score for each phrase, which is calculated using the English Wikipedia. At each iteration, the model first identifies the entity node(s) with the lowest degree or least amount of connected edges. Subsequently, if there are multiple candidates, we exploit the TF-IDF score to delete the node with the least frequency and the least centrality. The delete operation enables us to eliminate redundant branches for graph-level simplification.

**Replace.** To semantically simplify the graph, at each iteration, we select the most complex and least frequent word based on the IDF score. After selecting the complex word, we employ a two-stage approach to generate feasible replacements: (1) we obtain candidate phrases by leveraging a complex word dictionary from Simple PPDB [43], and (2) we consider a candidate phrase to be an appropriate substitution when it has a lower IDF score than the complex word.

For instance, suppose that the original KG contains the phrase: (*'The menu' 'offer' 'vegetarian alternatives'*). In that case, our model can replace the word *'alternatives'* with *'options'*. Similarly, for another phrase such as (*'Annual Avant Garde Festival of New York'*,

'inception', '1963'), we replace the word 'inception' with 'beginning'. These operations significantly enhance the semantic simplicity of the graphs.

**Merge.** To further simplify the graphs, we consider merging operations to certain types of edges:

- 1) Transitive edges: For two edges  $(e_1, r_1, e_2)$  and  $(e_2, r_2, e_3)$ , our model merges them into  $(e_1, r_1|sep|r_2, e_3)$ .  
For example, with *(Steven Gerrard, play, Liverpool F.C.)* and *(Liverpool F.C., locate in, the county of Merseyside)*, our model will merge them into *(Steven Gerrard, play|sep|locate in, the county of Merseyside)*
- 2) Edge pairs that share both head and tail entities: For two edges  $(e_1, r_1, e_2)$  and  $(e_1, r_2, e_2)$ , our model merges them into  $(e_1, r_1|sep|r_2, e_2)$ .  
For example, *(2013–14 Albanian Superliga, 'location', 'Albania')* and *(2013–14 Albanian Superliga, 'country', 'Albania')*.
- 3) Edge pairs that share the same head or tail entity: For two edges  $(e_1, r_1, e_2)$  and  $(e_1, r_2, e_3)$  or  $(e_2, r_1, e_1)$  and  $(e_3, r_2, e_1)$ , our model merges them into  $(e_2, r_1|sep|r_2, e_3)$ .

Furthermore, we use the sum of the IDF of all the phrases in the two edges to select the most complex and least informative pair to merge. The Merge operation improves our model's ability to further simplify the graph structure.

### 3.3 Generator

As part of the second phase of KGSimple, the proposed knowledge graph  $g_t$  is subjected to a generative KG-to-text model in each iteration, which produces a natural language output text  $y$ . This output text is generated based on the following probability formula:

$$P(y_{g_t}) = \prod_{i=1}^n P(w_i | w_1^{i-1}, g_t)$$

, where  $w_i$  is the token generated at position  $i$ ,  $n$  is the output length, and  $y_{g_t}$  is the output text corresponding to the proposed graph at iteration  $t$ .

By leveraging text generative models like GAP [8], our framework produces coherent and relevant sentences that describe the contents of the given graph. Note that multiple passes through this model, coupled with our scoring function, help improve the output of these models iteratively. This iterative process can reduce hallucination, which refers to the model generating content during the decoding step that lacks support from the given input data.

### 3.4 Scoring

Several previous studies have examined unsupervised TS, using heuristic-based reward functions to evaluate various syntactic and semantic aspects of the output text  $y$  based on individual metrics [18, 26, 27]. These reward functions aim to ensure that the output text is fluent ( $S_{fl}$ ), preserves the original meaning ( $S_{mp}$ ), and is simpler in content than the input ( $S_{sim}$ ), while also rejecting poor quality candidates generated through the iterative process. To refine these reward functions for the KG case, we propose using the product of all individual rewards to ensure balance across all criteria.

**3.4.1 Fluency.** To ensure that the generated candidates are grammatically correct and fluent, we use the syntactic log-odds ratio

(SLOR) score [42], which is defined as the sentence log-probability normalized by the unigram probability and sentence length, as the fluency score  $S_{fl}$ . The SLOR score works well as a proxy for fluency, as previous work has found that SLOR highly correlates with the human evaluation of grammatically acceptability [28] and fluency in the sentence compression task [20].

Given a unigram probability model, pre-trained language model (LM), and text, SLOR assigns a score to the text  $y_t$  at iteration  $t$  as:

$$SLOR(y_t) = \frac{1}{|y_t|} (\ln(p_{LM}(y_t)) - \ln(p_U(y_t))),$$

where  $|y_t|$  is the length of the text,  $p_{LM}(y_t)$  is the probability of  $y_t$  under a given LM,  $p_U$  is the unigram probability measures as:

$$p_U(y_t) = \prod_{w \in y_t} p(w)$$

, where  $p(w_i)$  is the unconditional probability of token  $w$  produced by a unigram-text model.

SLOR penalizes an LM's probability by the unigram probability, stabilizing the LM probability in the presence of rare tokens as the LM score may assign a lower probability to those texts containing rare tokens. This ensures that rare tokens do not penalize a text's fluency score which is important when evaluating the output text of a KG containing rare entity tokens as we do not want to penalize these rare tokens. Where previous works have used recurrent neural networks to calculate the LM probability for SLOR, we use a pre-trained GPT-2 [6] model with byte-pair encoding (BPE). To keep the score between [0,1] we avoid calculating the log of the model scores, instead opting to use their raw probabilities:

$$S_{fl}(y_t) = \frac{1}{|y_t|} (p_{LM}(y_t) - p_U(y_t))$$

**3.4.2 Saliency.** Saliency has been typically defined as the similarity between a candidate text and pseudo-reference text, where a text with high saliency has similar semantic meaning and both contain particularly important words [13]. To measure the saliency of a generated simplified text, we use the F1 of BERTScore [61] between the generated text at the current iteration  $y_t$  and the text generated from the original KG  $y_0$ , defined as:

$$S_{mp}(y_0, y_t) = \text{BERTScore}(y_0, y_t)$$

The BERTScore leverages the pre-trained BERT model when calculating the cosine similarity between two texts. Where previous work [11, 26] uses the cosine similarity as a hard filter, our saliency calculation acts as a soft score which is multiplied by our other scoring metrics, as the BERTScore typically falls between a very narrow high range in multiple settings [16].

**3.4.3 Simplicity.** Following previous work, we evaluate the simplicity  $S_{simp}$  of a generated text  $y$  with the Flesch Reading Ease (FRE) score [22], which measures the readability of a text based on the number of total sentences, words, and syllables. A lower score indicates that a text is more difficult to read. While the FRE score typically ranges from [0,100], the highest possible score is 121.22 with no theoretical lower bound. Therefore, to bound the range from [0, 1] we set the simplicity score to:

$$S_{si}(y_t) = \frac{FRE(y) - \lambda}{121.22 - \lambda},$$

where  $\lambda$  is a tunable hyper-parameter representing the lowest possible raw FRE score in a given dataset. For our purposes, we set  $\lambda = -30$ , in our experiments as a lower bound.

**3.4.4 Entity Score.** Unlike prior work in unsupervised TS, our task is a KG-to-text simplification task. Therefore, we check whether the generated text  $y_t$  accurately narrates the currently proposed KG  $g_t$ . To do so, we penalize the reward function if the KG-to-text model hallucinates any new entities that do not appear in the generated text. Because a KG is a compact and structured representation, it may be the case that the corresponding text would need to mention new entities in order to produce fluent sentences. We, therefore, impose a soft constraint if the model hallucinates new entities. The entity score  $S_{en}$  is defined as:

$$S_{en}(y_t, g_t) = 1 - \frac{|E_t^y \setminus E_t^g|}{|E_t^g|},$$

where  $E_t^g$  and  $E_t^y$  are the entities in  $g_t$  and  $y_t$ .

**3.4.5 Hard Constraints.** In order to prevent our model from producing invalid results, we define hard (binary) constraints on the proposed simplified KG and output text which cause the score to zero out if triggered.

As our sampled operations induce a reduction of the KG, our first hard constraint is a graph brevity penalty score  $S_{gb}$ , which is set to zero if the number of triples in proposed KG  $g_t$  is reduced below a certain threshold. In our experiments, we set the threshold as the ratio  $r_{op} = 0.6 \geq \frac{|T_{t-1}^g|}{|T_t^g|}$ , where  $|T_{t-1}^g|$  and  $|T_t^g|$  denote the number of triples in the previous and proposed graph, respectively.

Our second phase generates text from KG's, which may hallucinate entities that are likely to appear in a generated narration. Since our approach aims to remove unimportant entities' information, we verify that those entities that were specifically deleted from  $g$  do not appear in the simplified text. If these entities do appear in the generated text, the deleted entity score  $S_{de}$  is set to zero, automatically rejecting the sample.

**3.4.6 Overall.** As the various scores capture distinct facets of the simplified input, we compute the overall reward function as:

$$\begin{aligned} S(y_0, y_t, g_{t-1}, g_t) &= S_{fl}(y_t) \cdot S_{mp}(y_0, y_t) \\ &\cdot S_{si}(y_t) \cdot S_{en}(y_t, g_t) \\ &\cdot S_{gb}(g_{t-1}, g_t) \cdot S_{de}(y_t, g_0) \end{aligned}$$

## 3.5 Search Algorithms

In this paper, we investigate various acceptance conditions for the reward scoring function previously defined in our framework. These acceptance conditions include accepting the current proposal if and only if its score is greater than zero, accepting the proposal if its score is higher than the previous score, and employing an SA approach that promotes optimal solution exploration in the model's initial stages. The SA approach sets a threshold that is controlled by probability. If the candidate KG  $g_t$  is accepted at iteration  $t$ , it is processed next. However, if it is rejected, the algorithm proposes another operation on  $g_{t-1}$ .

While previous work has applied SA to other text generation tasks [30], we remodel SA for TS. In the SA approach, at each

iteration  $t$  a KG proposal  $g_t$  is accepted with probability:

$$\begin{aligned} p(A|g_{t-1}, g_t, y_{t-1}, y_t, T) &= \min\{1, e^{-\Delta E/T}\} \\ \Delta E &= S(y_t, g_0, g_t) - S(y_{t-1}, g_0, g_{t-1}), \end{aligned}$$

where  $T$  is the temperature or tolerance, initially set to a high value and decreased via a cooling rate [23].

## 4 EXPERIMENTS

### 4.1 Data

To evaluate the KGSimple framework, we use the WebNLG [15] KG-to-text dataset and DART [35] structured data record to text generation dataset, as the KGs contained within these datasets were constructed for natural language text generation. WebNLG is a hand-crafted triple-to-text dataset that contains graphs from DBpedia [5] with up to 7 triples paired with reference texts. DART is a human-annotated and automatic-converted dataset incorporated from WikiSQL [65], WikiTableQuestion [41], WebNLG 2017, and Cleaned E2E [38] with 21.6 words on average for each table.

We extract the more complex KGs from the WebNLG and DART datasets, considering those KGs containing more than 3 triples as complex. We thus experimented on 583 samples on WebNLG and 500 samples on DART. As there currently only exists unsupervised text-based approaches for TS, for a fair comparison we first generate a text with a KG-to-text model from each of the KGs. We use GAP [8] for WebNLG and T5 [44] for DART to generate such golden sentences. The generated text then serves as the complex text input in each of the baselines.

### 4.2 Competing Models

We evaluate KGSimple under the three aforementioned search criteria, namely: 1) greater than previous, 2) greater than zero, and the SA algorithm. As in previous work, we consider the complex input as an upper bound for each of the evaluation metrics. Each text generated by GAP/T5 acts as the golden reference, as it is the starting point used in all of the evaluated approaches.

For the unsupervised competing approaches, we compare them with state-of-the-art edit-based iterative approaches. First, we compare against [26], an iterative revision-based approach composed of a delete (RM), extraction (EX), lexical substitution (LS), and reordering (RO) operation. We include the RM+EX+LS+RO setting and denote it as EditTS. Next, we compare against USDP [56], a sentence simplification system that uses a dependency tree structure to decode a structurally simpler output. The generated output is then back-translated to English to generate lexical simplifications. Finally, we compare to GRS [11], an iterative framework that uses explicit edit operations to reduce complex text, including a generative paraphrase module. In our experiments, we use the PA+DL setting of GRS which performs both deletion and paraphrasing on an input text.

### 4.3 KGSimple Settings

For the KG-to-text generator, we use the GAP w/ type encoding [8] and T5 [35] models, which were fine-tuned on simple graphs (1-3 triples) from the WebNLG and DART dataset, respectively. GAP is a KG-to-text model that combines graph-aware elements into the BART [29] text generative model. We leave all of the default

**Table 1: Comparison of unsupervised text simplification models on WebNLG and DART. Len, CR, SC, and SR, refer to the Length, Compression Ratio, Syllable Count, and Syllable Ratio. EO, Add, and Delete, refer to Entity Text Overlap, Entity Text Added, and Entity Text Deleted. The constituency tree Height and Diameter are denoted by CTH and CTD, and the fluency (CoLA) and saliency (BERTScore) are denoted by F and S. GM is the geometric mean between the CR, SR, F, and S scores.  $\uparrow/\downarrow$  indicates that a higher/lower value is better.**

(a) Result on WebNLG												
	Len $\downarrow$	CR $\downarrow$	SC $\downarrow$	SR $\downarrow$	CTH $\downarrow$	CTD $\downarrow$	EO	Add $\downarrow$	Delete $\uparrow$	F $\uparrow$	S $\uparrow$	GM $\uparrow$
Baseline (GAP)	37.45	1.00	47.63	1.00	23.59	36.28	-	-	-	0.62	1.00	-
EditTS	33.17	0.90	40.82	0.89	22.74	33.73	0.89	0.29	0.53	0.48	<b>0.96</b>	0.27
USDP	24.78	0.69	<b>27.53</b>	<b>0.60</b>	20.92	<b>26.55</b>	0.59	<b>0.06</b>	1.26	0.54	0.93	0.50
GRS	26.04	0.71	31.87	0.68	<b>20.31</b>	29.51	0.75	0.68	1.39	0.56	0.93	0.47
Prev	31.39	0.85	38.70	0.82	22.22	32.51	0.75	0.53	1.26	0.63	0.93	0.35
SA	23.69	0.65	28.84	0.63	20.79	27.89	0.61	0.62	1.91	0.62	0.90	0.52
Zero-best	30.01	0.82	36.94	0.79	22.33	31.90	0.75	0.56	1.34	<b>0.64</b>	0.92	0.39
Zero-last	<b>23.07</b>	<b>0.64</b>	28.12	0.61	20.58	27.19	0.59	0.59	<b>1.91</b>	0.61	0.90	<b>0.53</b>

(b) Result on DART												
	Len $\downarrow$	CR $\downarrow$	SC $\downarrow$	SR $\downarrow$	CTH $\downarrow$	CTD $\downarrow$	EO	Add $\downarrow$	Delete $\uparrow$	F $\uparrow$	S $\uparrow$	GM $\uparrow$
Baseline (T5)	28.94	1.00	36.75	1.00	19.61	30.66	-	-	-	0.51	1.00	-
EditTS	26.94	0.94	33.03	0.91	19.48	29.63	0.67	0.18	0.25	0.43	0.96	0.22
USDP	18.21	0.67	<b>20.77</b>	<b>0.59</b>	18.06	<b>21.03</b>	0.30	<b>0.05</b>	1.13	0.37	0.93	0.46
GRS	21.78	0.75	27.25	0.75	17.69	25.91	0.53	0.45	0.90	0.51	0.94	0.42
Prev	26.58	0.93	33.02	0.90	18.85	29.03	0.58	0.16	0.46	0.51	<b>0.96</b>	0.24
SA	19.52	0.71	23.55	0.67	17.28	24.78	0.46	0.30	1.03	0.50	0.92	0.46
Zero-best	24.20	0.86	29.46	0.82	18.63	27.79	0.52	0.25	0.76	<b>0.52</b>	0.94	0.33
Zero-last	<b>17.72</b>	<b>0.64</b>	21.49	0.61	<b>16.81</b>	22.75	0.40	0.37	<b>1.20</b>	0.51	0.91	<b>0.51</b>

hyper-parameters for GAP and T5, and set the beam size to 5, with a repetition penalty of 1.0. T5 is a state-of-art data-to-text generation model and has been evaluated on DART. We use T5-base in our experiment and follow the same setting as in [44].

We set the probability of sampling the [delete, replace, merge] operations to a uniform distribution of  $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ . We run each input for 50 iterations with a batch size of 64. All components of our scorer are weighted equally, where we calculate the sentence probability for the fluency scorer using GPT-2 and calculate the BERTScore using the HuggingFace evaluate library<sup>2</sup>. For the *greater than previous* and *SA* search algorithms, we normalize all scores between 0 and 1. To extract entities from the output text in the entity scorer, we use the spaCy [34] named entity recognition module. All experiments were performed on 2 NVIDIA A-100 GPUs.

#### 4.4 Metrics

To evaluate all of the models, we measure intrinsic metrics associated with text simplicity, including the text length, number of syllables, semantic tree height, and semantic treewidth. As we propose a KG simplification approach, we also measure on entity-based metrics. These include counting the entity overlap between the input/output (entity text overlap), how many entities were added to

the output (entity text added), and how many entities were removed from the input (entity text deleted). As we want to ensure the output is coherent, we evaluate the *fluency* and *similarity* using the CoLA [57] and BERTScore. We use CoLA, as the LM perplexity does not account for repetition or grammar when scoring. For CoLA, we report the probability that a text is indeed acceptable. We exclude the FKGL from our evaluation, as FKGL was designed for a text of at least 200 words [4, 58]. Additionally, Tanprasert and Kauchak [50] have recently found that FKGL can be easily manipulated via basic post-processing steps and argue to instead report the individual components of FKGL, i.e., sentence length and syllable count. Therefore, we report these, along with coherence metrics (fluency and saliency), and a geometric mean which combines the size metrics with the coherence metrics. Unlike prior work on unsupervised TS, where the data still comes from a supervised dataset, our datasets do not contain any simplified text. We, therefore, exclude calculating the BLEU [40] or SARI [60] scores.

## 5 RESULTS

We present the results on WebNLG and DART in table 1. We refer to the complex generated text data as *Baseline*. For the KGSimple search algorithm with the "greater than zero" condition, we report

<sup>2</sup><https://github.com/huggingface/evaluate>

both the last accepted iteration (*Zero-last*) and the best-scored iteration (*Zero-best*), as this condition is more loosely constrained than our other conditions. We denote the "greater than prev" and SA algorithms as *Prev* and *SA*, respectively.

## 5.1 WebNLG

From table 2a, we can observe that KGSimple with SA outperforms the KGSimple configuration with the *Prev* search constraint in terms of intrinsic simplification metrics, with a text compression ratio of 0.65, syllable ratio of 0.63, and constituency tree height/diameter of 20.79 and 27.89. Although *Zero-last* produces slightly shorter sentences than the *SA* approach, the *SA* approach produces more fluent sentences. Conversely, the *Zero-best* approach and the more tightly conditioned *Prev* method produce more fluent sentences. Hence, for the unsupervised text simplification approaches there may be some trade-off between length and fluency which future work can further study. When scoring the geometric mean between the length ratios, fluency, and saliency, we find that the *SA* and *Zero-last* search constraints perform similarly, where *Zero-last* produces slightly shorter texts and *SA* produces slightly more fluent texts. In terms of text entity metrics, the *SA* and *Zero* approaches perform similarly, where *Prev* keeps more of the entities from the original text. All KG-based simplification approaches perform consistently and similarly for fluency and saliency, where they seem to be able to produce grammatically correct and semantically similar texts. This may be in part because the generator excels at producing coherent text from KGs, while our simplification approach manages to keep and combine the important KG features.

Compared to the text-based unsupervised models, KGSimple can generate shorter texts, while producing similar results in terms of syllable counts and constituency tree height/diameter. KGSimple also produces more fluent texts compared to GRS and USDP. While the model proposed by Kumar et al. [26] has similar fluency scores compared to KGSimple, their text lengths do not decrease by much, with a compression ratio of 0.85, compared to the KGSimple *SA* compression ratio of 0.65 and *Zero-last* compression ratio of 0.64. For the entity-based metrics, KGSimple deletes more entities than the text-based approach, but can also introduce some newer entities. When compared to USDP and GRS, KGSimple has similar text entity overlap. The KG-based approaches all outperform the text-based approach in linguistic acceptability (fluency). Note that although GRS uses the CoLA as a constraint in their search algorithm, KGSimple still outperforms GRS on this metric. While GRS is the only other generative-based unsupervised approach, KGSimple consistently outperforms the model. Therefore, KGSimple appears to be a better overall system when desiring more variability and flexibility in simplifying sentences. The text-based approaches do achieve a slightly higher BERTScore than the KG-based approaches. However, this may be because the edit text-based approaches focus on directly modifying (cutting) the original text, while our KG-based approach needs to produce a new text from the currently proposed KG at each iteration.

## 5.2 DART

From table 1b, compared to the results from WebNLG, we can observe a consistent performance of KGSimple on DART. With

**Table 2: KGSimple output graph analysis: number of triples (Triples), in/out triple ratio (TR), entity ratio (ER), and relation ratio (RR).**

(a) Result on WebNLG				
	Triples	TR	ER	RR
Prev	3.31	0.67	0.84	0.74
SA	2.09	0.45	0.66	0.50
Zero	2.03	0.44	0.65	0.49
(b) Result on DART				
	Triples	TR	ER	RR
Prev	4.32	0.78	0.87	0.83
SA	2.19	0.41	0.61	0.45
Zero	2.03	0.39	0.58	0.42

**Table 3: Overall ratio analysis for the accepted individual individual Delete (D), Replace (R), and Merge (M) graph operations for KGSimple.**

(a) Result on WebNLG			
	D	R	M
Prev	0.33	0.34	0.33
SA	0.26	0.45	0.28
Zero	0.26	0.47	0.28
(b) Result on DART			
	D	R	M
Prev	0.55	0.34	0.11
SA	0.50	0.33	0.17
Zero	0.47	0.35	0.18

SA, KGSimple achieves better performance on the intrinsic simplification metrics (length and syllable count), compared to *Prev*. On the other hand, the method with a tighter condition, *Zero-best* compared to *Zero-last*, again generated more fluent sentences. This provides further evidence of the possible tradeoff between the length and fluency of text-simplification methods. Generally, all KG-based methods achieve similar performance for fluency and saliency, while *Prev* preserves more entities from the original text as in WebNLG. Again here, as with WebNLG, the *Zero-last* and *SA* approaches gave the best geometric means between compression ratio, syllable ratio, fluency, and saliency scores.

On DART, unlike in the experiments on WebNLG, USDP generates shorter texts than those generated by the KGSimple *SA* algorithm. However, note the discrepancy in the fluency score, where USDP has the lowest linguistically acceptable generated texts. So while the texts may be the shortest, they may not be as useful as those generated by KGSimple, which may have a slightly better balance between the metrics. While EditTS and GRS achieve compatibly similar fluency performance against KGSimple, KGSimple *Zero-last* and *SA* generate shorter texts with lower compression ratios of 0.64 and 0.71. On the entity-based evaluation, KGSimple adds fewer new entities while deleting more entities and less entity overlap than all text-based approaches except USDP. This may be because USDP directly removes sentences/clauses from the text. On linguistic metrics, KGSimple, on the other hand, generates more fluent texts with the highest fluency score of 0.52 and the highest saliency score of 0.96. Additionally, compared to the generative-based approach GRS, our *SA* and *Zero-last* KGSimple approach generates shorter sentences on average, while maintaining a consistent text fluency score.

Generally, the performance of KGSimple on DART is consistent with the result on WebNLG when compared to the text-based baselines. Compared to the text-based baselines which rely on using

CT	ST
the bronze ataturk monument (izmir) is located in izmir and was inaugurated on 27 july 1932. the capital of the country is ankara and the largest city istanbul. the country's leader is ahmet davutoğlu and the currency is the turkish lira.	the bronze atatürk monument is located in the largest city of izmir and was inaugurated on the 27th of july, 1932.
the american submarine nr-1 has a 3.8m ship beam and a top speed of 8.334 km/h. it was built by the company general dynamics electric boat and was launched on january 25, 1969.	the american submarine nr-1 has a top speed of 8.334 km/h and was launched on january 25, 1969.
the city of aarhus, which has to its northeast mols, is governed by a magistrate. the city is the location of the school of business and social sciences at the aarhus university which is affiliated with the european university associations which has its headquarters in brussels	the school of business and social sciences at the aarhus university has its headquarters in the city of aarhus and has strong connections with the academic community.
the city of lahore is served by the allama iqbal international airport which is located in pakistan. the airport is operated by the pakistan civil aviation authority and has a runway length of 2900.0.	the city of lahore is served by the allama iqbal international airport which is operated by the civil aviation authority.
the 14th new jersey volunteer infantry monument is located in the district of the monocacy national battlefield, frederick county, maryland, united states. the monument was established on 11th july 1907 and belongs to the category of historic districts in the united states.	the 14th new jersey volunteer infantry monument is located in frederick county, maryland, united states.

**Table 4: Examples from the WebNLG and DART datasets. Left: complex text (CT). Right: simplified text (ST) results generated by the KGSimple framework.**

separate models for text generation and simplification, KGSimple integrates the KG-to-text models in its simplification framework, to produce syntactically simpler and more fluent texts, while keeping fewer entities from the original sentences and introduces fewer entities for text generation.

## 6 ANALYSIS

### 6.1 Graphs

As KGSimple is a graph-centric approach, we compare the input KGs to the reduced KGs from which the simplified text is generated in table 2. We compare the graph statistics in each of our search settings, including the number of triples (Triples), output-to-input KG size ratio (TR), output-to-input unique entity ratio (ER), and output-to-input unique relation ratio (RR). From the table, we can see that although the simulated annealing (SA) algorithm imposed a tighter constraint than the *Zero* algorithm, which accepts the current proposal as long as the overall score was greater than zero, the SA was able to reduce the KG almost as close to the *Zero-last* condition. In contrast, as expected, the *Prev* algorithm has the overall largest number of final triples and restricts the graph reduction to 67% and 78% of the original KG. Thus, our input-output graph analysis confirms that the SA condition can encourage further KG reduction (search space exploration) for KG-text simplification, while the more restrictive *Prev* condition may get stuck at local minima. This is reflected when analyzing both the entity (ER) and relation (RR) analysis, where the *Prev* condition contains a larger ratio compared to SA and *Zero*.

### 6.2 Operations

The KGSimple framework is an iterative approach that proposes changes to a KG based on sampled graph-reduction operations. Table 3 compares the relative acceptance ratios of the delete, replace, and merge operations. From the results on WebNLG, in all search algorithms, our most accepted operation was *replace*, while *delete* and *merge* were not accepted as often. For DART, the most accepted operation was *delete*, while *merge* was not as frequently accepted as in the WebNLG dataset. We currently use node centrality and TD-IDF to find those nodes to delete from the KG. While these nodes may not be as important in the KG, they may at times convey information that is crucial when transcribing the text either for semantic or fluency purposes. The discrepancy in both datasets for *merge* may be because the datasets contain different ontologies, graph structures, and relations. WebNLG may contain more sets of triples that are acceptable to merge, while DART's KGs may contain a more sporadic structure. The *merge* operation may be further modified depending on the overall structure of the KG's and ontology found in the specific dataset.

### 6.3 Examples

We showcase five examples produced by KGSimple in Table 4 by comparing the text produced by the KG-to-text model on the original complex KG to the last accepted output text which was generated from the simplified KG. From these examples, we see that KGSimple has the capacity to simplify text by implicitly cutting sentences/clauses, replacing tokens, and merging phrases via the KG



**Table 5: Case study from the WebNLG. Show the original text, simplified result by USDP, and result by KGSimple framework step by step**

ORIGINAL TEXT:
The alternative name for the AMC Matador is Vam Classic and it is classed as a mid-size car. It is made in Kenosha, Wisconsin, and has an AMC straight 6 engine.
USDP:
and it is classed - <mask> as a mid - size car .
KGSIMPLE:
<b>Graph 0:</b> (Vam classic, <b>alternative name</b> , AMC Matador), (Kenosha, Wisconsin, assembly, AMC Matador), (mid-size car, class, AMC Matador), (AMC straight-6 engine, engine, AMC Matador)
↓
<b>Graph 1:</b> (Kenosha, Wisconsin, <b>assembly</b> , AMC Matador), (mid-size car, class, AMC Matador), (AMC straight-6 engine, engine, AMC Matador)
↓
<b>Graph 2:</b> (Kenosha, Wisconsin, found, AMC Matador), (mid-size car, class, AMC Matador), ( <b>AMC straight-6 engine</b> , engine, AMC Matador)
↓
<b>Graph 3:</b> (Kenosha, Wisconsin, found, AMC Matador), (mid-size car, class, AMC Matador)
↓
<b>Simplified text:</b> The AMC Matador is classed as a mid-size car and is found in Kenosha, Wisconsin.

reduction operations. For example in the second example, where the text is about the "submarine nr-1", our approach removes phrases regarding the "ship beam" and "general dynamics electric boat", which were nodes in the complex KG. KGSimple also merges the two sentences via the merge operation on the KG. However, as our method is a generative approach (KG-to-text), one limitation is that the approach is prone to hallucination, similar to other generative approaches, such as GRS [11]. For example, in the first example referring to the "bronze ataturk monument", the simplified text implies that the monument is located in the largest city of "izimir", but the original text describes "istanbal" as the largest city. Second, replacement is limited by the dictionary contained in PPDB, as some complex words are not substituted. Nevertheless, as seen from all the examples, our approach can robustly cut out sentences, clauses, and specific details (entities/reactions) which are not as important according to KGSimple's reduction step. For instance, in the last example about the "14th new jersey volunteer infantry monument", KGSimple simplifies some information about the exact location of the monument while fluently conveying its general location. Overall, the results across both the WebNLG and DART complex KGs show that KGSimple can coherently simplify the information found in the KGs and transform them into natural language sentences.

## 6.4 Case Study

To further demonstrate how KGSimple simplifies text by leveraging the graph compared to the text-based approaches, we show a

detailed simplification in this section and compare it to the USDP model. As shown in Table 5, the original sentence is:

**The alternative name for the AMC Matador is Vam Classic and it is classed as a mid-size car. It is made in Kenosha, Wisconsin, and has an AMC straight 6 engine.**

Note, that as the original texts are all generated by the GAP KG-to-text model, there may be discrepancies between the graph and the text. For example, here, GAP replaced **assembly** in the graph with **made** when generating the original text.

We can see that the simplified text generated by USDP is shorter than KGSimple, but is less fluent and informative. In fact, USDP is more of a sentence-level method which usually keeps a sub-sentence as the simplified result. Compared to such an approach, KGSimple is able to simplify texts on various granularities, including the word level, by leveraging triple-level operations. Moreover, instead of selecting individual sentences, KGSimple also attempts to merge sentences together for further simplification via its merge operations on the KG. Following the operations also improves the interpretability of the simplification process.

Specifically, in step 1, KGSimple first identifies **alternative name** as an uncommon term and removes the related triple to simplify the graph. Then in step 2, KGSimple locates the term **assembly** as a complex term with replacement in the dictionary. Thus, it replaces the word with **found** to simplify the graph on the text aspect. Finally, in step 3, the model notices the term **AMC straight-6 engine** with the lowest TF-IDF score in the graph. Again, the delete operation behaves to remove this triple to further simplify the graph. As there is no further operation applied to the graph, GAP translates the graph back into the text as the final result. This example demonstrates how KGSimple simplifies texts on both the graph and text level, with explainable progress at each operation, which further shows the usefulness of KGs when attempting to interpret text simplification models.

## 7 CONCLUSION

We proposed KGSimple, a novel unsupervised KG-centric text simplification framework. We have demonstrated that text simplification can be accomplished via a KG-first approach, where the KG is reduced to its more crucial components and then narrated to natural language text via KG-to-text models. By considering both the KG and text in the reward, our approach has the potential to outperform text-centric unsupervised models as indicated by our results on the WebNLG and DART datasets, specifically able to generate less complex sentences while maintaining a relatively high fluency score. As we have developed the first framework for KG-text simplification, we encourage future work in other natural language generation tasks to explore KG-oriented approaches. We also encourage adapting our framework to other types of structured data such as tables or relational databases, where consumer healthcare data such as electronic medical records are often stored.

## ACKNOWLEDGEMENTS

This work is partially supported by the Arnold and Lisa Goldberg Endowed Professorship and DARPA under Award #FA8750-18-2-0014 (AIDA/GAIA).

## REFERENCES

- [1] Emil Abrahamsson, Timothy Forni, Maria Skeppstedt, and Maria Kvist. 2014. Medical text simplification using synonym replacement: Adapting assessment of word difficulty to a compounding language. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*. 57–65.
- [2] Sweta Agrawal, Weijia Xu, and Marine Carpuat. 2021. A Non-Autoregressive Edit-Based Approach to Controllable Text Simplification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 3757–3769. <https://doi.org/10.18653/v1/2021.findings-acl.330>
- [3] Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning How to Simplify From Explicit Labeling of Complex-Simplified Text Pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 295–305. <https://aclanthology.org/I17-1030>
- [4] Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoit 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*. Association for Computational Linguistics, Online, 4668–4679. <https://doi.org/10.18653/v1/2020.acl-main.424>
- [5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11–15, 2007. Proceedings*. Springer, 722–735.
- [6] 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 (2020), 1877–1901.
- [7] Ping Chen, John Rochford, David N Kennedy, Soussan Djamasbi, Peter Fay, and Will Scott. 2017. Automatic text simplification for people with intellectual disabilities. In *Artificial Intelligence Science and Technology: Proceedings of the 2016 International Conference (AIST2016)*. World Scientific, 725–731.
- [8] Anthony Colas, Mehrdad Alvandipour, and Daisy Zhe Wang. 2022. GAP: A Graph-aware Language Model Framework for Knowledge Graph-to-Text Generation. In *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 5755–5769. <https://aclanthology.org/2022.coling-1.506>
- [9] Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 2021. Event-Narrative: A large-scale Event-centric Dataset for Knowledge Graph-to-Text Generation. *arXiv preprint arXiv:2111.00276* (2021).
- [10] Tanvi Dadu, Kartikey Pant, Seema Nagar, Ferdous Barbhuiya, and Kuntal Dey. 2021. Text Simplification for Comprehension-based Question-Answering. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*. 1–10.
- [11] Mohammad Dehghan, Dhruv Kumar, and Lukasz Golab. 2022. GRS: Combining Generation and Revision in Unsupervised Sentence Simplification. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics, Dublin, Ireland, 949–960. <https://doi.org/10.18653/v1/2022.findings-acl.77>
- [12] Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An Neural Programmer-Interpreter Model for Sentence Simplification through Explicit Editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 3393–3402. <https://doi.org/10.18653/v1/P19-1331>
- [13] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research* 22 (2004), 457–479.
- [14] Erwin Filtz. 2017. Building and processing a knowledge-graph for legal data. In *The Semantic Web: 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part II 14*. Springer, 184–194.
- [15] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating Training Corpora for NLG Micro-Planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 179–188. <https://doi.org/10.18653/v1/P17-1017>
- [16] Michael Hanna and Ondřej Bojar. 2021. A Fine-Grained Analysis of BERTScore. In *Proceedings of the Sixth Conference on Machine Translation*. Association for Computational Linguistics, Online, 507–517. <https://aclanthology.org/2021.wmt-1.59>
- [17] Xuanli He, Quan Hung Tran, Gholamreza Haffari, Walter Chang, Zhe Lin, Trung Bui, Franck Dernoncourt, and Nhan Dam. 2020. Scene Graph Modification Based on Natural Language Commands. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 972–990. <https://doi.org/10.18653/v1/2020.findings-emnlp.87>
- [18] Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14, 8 (2002), 1771–1800.
- [19] Alexander Miserlis Hoyle, Ana Marasović, and Noah A. Smith. 2021. Promoting Graph Awareness in Linearized Graph-to-Text Generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 944–956. <https://doi.org/10.18653/v1/2021.findings-acl.82>
- [20] Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-Level Fluency Evaluation: References Help, But Can Be Spared!. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Brussels, Belgium, 313–323. <https://doi.org/10.18653/v1/K18-1031>
- [21] Pei Ke, Haozhe Ji, Yu Ran, Xin Cui, Liwei Wang, Linfeng Song, Xiaoyan Zhu, and Minlie Huang. 2021. JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 2526–2538. <https://doi.org/10.18653/v1/2021.findings-acl.223>
- [22] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel*. Technical Report. Naval Technical Training Command Millington TN Research Branch.
- [23] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
- [24] Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 2284–2293. <https://doi.org/10.18653/v1/N19-1238>
- [25] Reno Kriz, João Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. 2019. Complexity-Weighted Loss and Diverse Reranking for Sentence Simplification. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3137–3147. <https://doi.org/10.18653/v1/N19-1317>
- [26] Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative Edit-Based Unsupervised Sentence Simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7918–7928. <https://doi.org/10.18653/v1/2020.acl-main.707>
- [27] Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A. Hearst. 2021. Keep It Simple: Unsupervised Simplification of Multi-Paragraph Text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 6365–6378. <https://doi.org/10.18653/v1/2021.acl-long.498>
- [28] Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive science* 41, 5 (2017), 1202–1241.
- [29] 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*. Association for Computational Linguistics, Online, 7871–7880. <https://doi.org/10.18653/v1/2020.acl-main.703>
- [30] Jingjing Li, Zichao Li, Lili Mou, Xin Jiang, Michael Lyu, and Irwin King. 2020. Unsupervised text generation by learning from search. *Advances in Neural Information Processing Systems* 33 (2020), 10820–10831.
- [31] Mounica Maddela, Fernando Alva-Manchego, and Wei Xu. 2021. Controllable Text Simplification with Explicit Paraphrasing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 3536–3553. <https://doi.org/10.18653/v1/2021.naacl-main.277>
- [32] Louis Martin, Éric de la Clergerie, Benoit Sagot, and Antoine Bordes. 2020. Controllable Sentence Simplification. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 4689–4698. <https://aclanthology.org/2020.lrec-1.577>
- [33] Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoit Sagot. 2022. MUSS: Multilingual Unsupervised Sentence Simplification by Mining Paraphrases. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*. European Language Resources Association, Marseille, France, 1651–1664. <https://aclanthology.org/2022.lrec-1.176>
- [34] Ines Montani, Matthew Honnibal, Matthew Honnibal, Sofie Van Landeghem, Adriane Boyd, Henning Peters, Paul O’Leary McCann, jim geoverdi, Jim O’Regan, Maxim Samsonov, György Orosz, Daniël de Kok, Marcus Blättermann, Duygu Altınok, Søren Lind Kristiansen, Madeesh Kannan, Raphael Mitsch, Raphaël

- Bourhonesque, Edward, Lj Miranda, Peter Baumgartner, Richard Hudson, Exchusion Bot, Roman, Leander Fiedler, Ryn Daniels, Wannaphong Phatthiyaphibun, Grégory Howard, and Yohei Tamura. 2023. *explosion/spaCy: v3.5.1: spancat for multi-class labeling, fixes for textcat+transformers and more*. <https://doi.org/10.5281/zenodo.7715077>
- [35] Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-Domain Structured Data Record to Text Generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 432–447. <https://doi.org/10.18653/v1/2021.naacl-main.37>
- [36] Shashi Narayan and Claire Gardent. 2016. Unsupervised Sentence Simplification Using Deep Semantics. In *Proceedings of the 9th International Natural Language Generation conference*. Association for Computational Linguistics, Edinburgh, UK, 111–120. <https://doi.org/10.18653/v1/W16-6620>
- [37] Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring Neural Text Simplification Models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, 85–91. <https://doi.org/10.18653/v1/P17-2014>
- [38] Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. The E2E Dataset: New Challenges For End-to-End Generation. In *18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 201–206.
- [39] Kostiantyn Omelianchuk, Vipul Raheja, and Oleksandr Skurzhanskyi. 2021. Text Simplification by Tagging. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, Online, 11–25. <https://aclanthology.org/2021.bea-1.2>
- [40] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. <https://doi.org/10.3115/1073083.1073135>
- [41] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1470–1480.
- [42] Adam Pauls and Dan Klein. 2012. Large-Scale Syntactic Language Modeling with Treelets. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, 959–968. <https://aclanthology.org/P12-1101>
- [43] Ellie Pavlick and Chris Callison-Burch. 2016. Simple PPDB: A Paraphrase Database for Simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, 143–148. <https://doi.org/10.18653/v1/P16-2024>
- [44] Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. Investigating Pretrained Language Models for Graph-to-Text Generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*. Association for Computational Linguistics, Online, 211–227. <https://doi.org/10.18653/v1/2021.nlp4convai-1.20>
- [45] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. 2017. Learning a health knowledge graph from electronic medical records. *Scientific reports* 7, 1 (2017), 1–11.
- [46] Martin Schmitt, Leonardo F. R. Ribeiro, Philipp Dufter, Iryna Gurevych, and Hinrich Schütze. 2021. Modeling Graph Structure via Relative Position for Text Generation from Knowledge Graphs. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*. Association for Computational Linguistics, Mexico City, Mexico, 10–21. <https://doi.org/10.18653/v1/2021.textgraphs-1.2>
- [47] Sanja Štajner. 2021. Automatic Text Simplification for Social Good: Progress and Challenges. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Association for Computational Linguistics, Online, 2637–2652. <https://doi.org/10.18653/v1/2021.findings-acl.233>
- [48] Sanja Štajner and Maja Popović. 2016. Can text simplification help machine translation?. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*. 230–242.
- [49] 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*. Association for Computational Linguistics, Florence, Italy, 2058–2068. <https://doi.org/10.18653/v1/P19-1198>
- [50] Teerapaun Tanprasert and David Kauchak. 2021. Flesch-Kincaid is Not a Text Simplification Evaluation Metric. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*. Association for Computational Linguistics, Online, 1–14. <https://doi.org/10.18653/v1/2021.gem-1.1>
- [51] Hoang Van, Zheng Tang, and Mihai Surdeanu. 2021. How May I Help You? Using Neural Text Simplification to Improve Downstream NLP Tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 4074–4080. <https://doi.org/10.18653/v1/2021.findings-emnlp.343>
- [52] Laurens Van den Bercken, Robert-Jan Sips, and Christoph Lofi. 2019. Evaluating neural text simplification in the medical domain. In *The World Wide Web Conference*. 3286–3292.
- [53] Peter JM Van Laarhoven, Emile HL Aarts, Peter JM van Laarhoven, and Emile HL Aarts. 1987. *Simulated annealing*. Springer.
- [54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=rJXMpikCZ>
- [56] Vy Vo, Weiqing Wang, and Wray Buntine. 2022. Unsupervised Sentence Simplification via Dependency Parsing. *arXiv preprint arXiv:2206.12261* (2022).
- [57] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics* 7 (2019), 625–641.
- [58] Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics* 3 (2015), 283–297.
- [59] 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 (2016), 401–415. [https://doi.org/10.1162/tacl\\_a\\_00107](https://doi.org/10.1162/tacl_a_00107)
- [60] 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 (2016), 401–415.
- [61] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. [n. d.]. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.
- [62] 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*. Association for Computational Linguistics, Copenhagen, Denmark, 584–594. <https://doi.org/10.18653/v1/D17-1062>
- [63] Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018. Integrating Transformer and Paraphrase Rules for Sentence Simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3164–3173. <https://doi.org/10.18653/v1/D18-1355>
- [64] 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*, Vol. 34. 9668–9675.
- [65] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* (2017).