
Unsupervised Text Generation by Learning from Search

Jingjing Li^{1*}, Zichao Li^{2*}, Lili Mou³, Xin Jiang², Michael R. Lyu¹, Irwin King¹

¹The Chinese University of Hong Kong ²Huawei Noah's Ark Lab

³University of Alberta; Alberta Machine Intelligence Institute (Amii)

{lijj,lyu,king}@cse.cuhk.edu.hk

{li.zichao,jiang.xin}@huawei.com

doublepower.mou@gmail.com

Abstract

In this work, we present TGLS, a novel framework to unsupervised Text Generation by Learning from Search. We start by applying a strong search algorithm (in particular, simulated annealing) towards a heuristically defined objective that (roughly) estimates the quality of sentences. Then, a conditional generative model learns from the search results, and meanwhile smooth out the noise of search. The alternation between search and learning can be repeated for performance bootstrapping. We demonstrate the effectiveness of TGLS on two real-world natural language generation tasks, paraphrase generation and text formalization. Our model significantly outperforms unsupervised baseline methods in both tasks. Especially, it achieves comparable performance with the state-of-the-art supervised methods in paraphrase generation.

1 Introduction

Text generation refers to a wide range of tasks involving generating natural language, including but not limited to machine translation [20, 21, 19], sentence simplification [30, 42], and text summarization [4, 1]. Recent success of neural-based text generation relies heavily on a large parallel dataset for training, which may not be available in real-world natural language processing (NLP) applications. In this work, we consider unsupervised text generation, where no parallel data is available. This setting is more challenging, and has significant potential in both scientific research (e.g., low-resource language processing) and industrial applications (e.g., cold start for a new NLP application).

Early work tackles unsupervised text generation by rules or templates [45, 28]. While such approaches do not require parallel corpora, the generated sentences are highly subject to the rules, and hence lack the flexibility of natural language. Other work constructs pseudo-parallel data, which is only feasible for certain tasks like unsupervised machine translation [20].

Recently, researchers have developed search-based techniques for unsupervised text generation [29, 18, 38, 24], where a heuristically defined scoring function evaluates the quality of a sentence, involving language fluency, semantic compliance, and other task-specific aspects. Then, the algorithm performs word-level edits (such as word deletion, insertion, and replacement) to search towards a (possibly local) optimum of the scoring function. With a reasonably designed scoring function, such approaches are shown to be effective in a variety of applications like paraphrase generation [29, 24], sentence summarization [38], and text simplification [18].

However, the search-based approach has two major drawbacks: 1) The inference efficiency is low. To obtain an output sentence, the search algorithm would perform a few hundred steps of local edits and

*equal contribution

re-evaluations. This could be considerably slower than an autoregressive decoder, which generates words sequentially. 2) The search could yield noisy results, since the scoring function is defined heuristically and the search is conducted locally in a discrete sentence space.

To this end, we propose a new framework for unsupervised Text Generation by Learning from Search (TGLS), which contains a strong search module that explores the sentence space, as well as a learning module that learns from the search results. For the search module, we adopt the simulated annealing (SA) algorithm. At each step, SA proposes a local edit by a neural network, and then either accepts or rejects the proposal based on a heuristically defined scoring function. For learning, we employ two methods to train the conditional generative model, word-level cross-entropy loss and the sequence-level max-margin loss. Within TGLS, the search and learning can be boosted by each other in an iterative fashion. That is, the search results serve as the pseudo-reference for training the conditional generator, which in turn benefits SA search by serving as a more meaningful initial state. As for implementation, TGLS involves two pretrained language models: a) the uni-directional GPT2 [34], which is suitable for likelihood-based fluency evaluation and conditional generation; and b) the bi-directional RoBERTa [25], which is better at semantic evaluation and contextual word-level prediction.

The main contributions of our paper include: 1) We propose TGLS, a generic learning-from-search framework for unsupervised text generation. 2) We demonstrate efficient methods of incorporating the large-scale pretrained language models into our TGLS framework. 3) We conducted experiments on two different tasks: paraphrasing and text formalization. In both experiments, TGLS significantly outperforms unsupervised baseline methods. Moreover, TGLS achieves comparable performance to recent supervised models [7] in the paraphrasing task. 4) For text formalization (an example of text style transfer), we are also the first to design a search-based method, and further extend it into the proposed TGLS framework.

2 Approach

Our TGLS framework involves two stages of search and learning. In the first stage, we perform simulated annealing (SA) search [24] and treat the obtained output sentences as pseudo-references. Then, we train an autoregressive GPT2 as the text generator [34] by word-level cross-entropy (CE) supervised learning, which enables our model to learn quickly. In the second stage, the search is conducted in a hybrid approach: the GPT2 executes beam search and the outputs are taken as the initial state of the SA algorithm again for iterative performance improvement. Later, we perform max-margin (MM) learning to better distinguish between higher-scored sentences and other high-probability but sub-optimal sentences. Figure 1 provides an overview of the two stages of search and learning in TGLS.

2.1 Simulated Annealing Search

The search-based text generation [29, 24] relies on a heuristic-based objective function $s(y|x)$ that (roughly) evaluates the quality of an output sequence y given the input x (usually, one or a few sentences). Typically, the objective involves language modeling fluency $s_{lm}(x)$, semantic compliance $s_{semantic}(x, y)$, and other task-specific scorers $s_{task}(y, \cdot)$. These individual scorers are combined by the product of experts [13]:

$$s(y|x) = s_{lm}(y) \cdot s_{semantic}(x, y) \cdot s_{task}(y, \cdot) \quad (1)$$

We adopt simulated annealing (SA) [16, 24], which performs local stochastic search to maximize the objective. Concretely, SA starts from an initial candidate output sentence $y^{(0)}$, which is set to the input x in our first-stage SA. For the second stage, it will be the output of our GPT2 model.

At a search step t , SA iteratively proposes a new candidate y' by local edits of $y^{(t)}$, namely, word insertion, deletion, and replacement. The proposal y' is accepted with probability $p(\text{accept}|y', y^{(t)}, x, T) = \min\{1, \exp(\frac{s(y'|x) - s(y^{(t)}|x)}{T})\}$. Then, $y^{(t+1)} = y'$ if y' is accepted, or otherwise, $y^{(t+1)} = y^{(t)}$. In SA, T is a temperature controlling how greedy the search algorithm is. Usually, T is high at the beginning of search so as to be more explorative, and then T is cooled down to achieve a better (local) optimum. Although we follow the generic SA framework of text generation as in [24], the objective function and proposal are largely redesigned, detailed below.

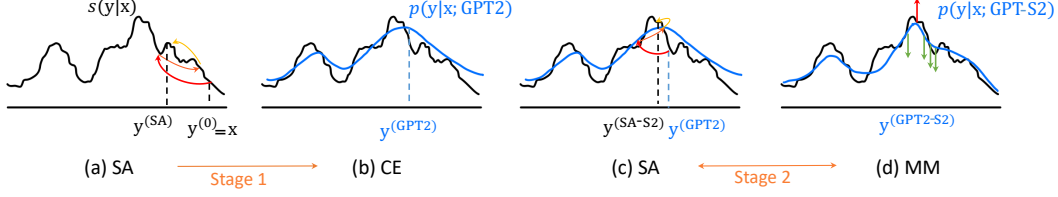


Figure 1: Overview of TGLS. (a) First-stage search by simulated annealing (SA). (b) First-stage learning by cross-entropy (CE) loss. (c) Second-stage search by SA. (d) Second-stage learning by max-margin (MM) loss. The horizontal axis represents the sentence space.

Fluency scorer (s_{lm}). The fluency of a sentence can oftentimes be approximated by a language model’s predicted probability. Previous search-based work uses recurrent neural networks for fluency evaluation [29, 24]. In our work, we use the large-scale pretrained GPT2 model [34]. For an output $y = y_1 \cdots y_n$, the language fluency scorer is the joint likelihood of y , given by $s_{lm}(y) = (\prod_{i=1}^n p(y_i | y_1, \dots, y_{i-1}))^\alpha$, where α is a hyperparameter balancing s_{lm} with other scorers in (1). In fact, we use the vocabulary of GPT2 with byte-pair encoding (BPE), and y_i here is a token after BPE segmentation. Our GPT2 is fine-tuned with non-parallel in-domain corpora to learn the specificity of a task.

Semantic scorer ($s_{semantic}$). In this part, we extend the semantic scorers in [24] with a RoBERTa [25]. Fine-tuning details are presented in Appendix A. Compared with autoregressive GPT2 used for fluency evaluation, RoBERTa is pretrained by masked language modeling, and is better at feature representation. Let $x = (x_1, \dots, x_m)$ be a sentence. RoBERTa computes a contextualized representation of a word in the sentence as $\text{RoBERTa}(x_i, x)$.

A word-level semantic scorer evaluates how much keyword information (detected by Rake [37]) is preserved, given by the least matched keyword of x :

$$s_{word}(y, x) = \min_{k \in \text{keyword}(x)} \max_{y_i \in Y} \text{RoBERTa}(k, x)^\top \text{RoBERTa}(y_i, y) \quad (2)$$

A sentence-level semantic scorer evaluates the cosine similarity of two sentence vectors $s_{sent}(y, x) = \frac{\mathbf{y}^\top \mathbf{x}}{\|\mathbf{y}\| \|\mathbf{x}\|}$, where the sentence vector is given by the RoBERTa feature of the padded token [BOS] at the beginning end of a sentence, i.e., $\mathbf{x} = \text{RoBERTa}([\text{BOS}], x)$ and \mathbf{y} is computed analogously.

Finally, the semantic scorer is the product of both word- and sentence-level scores as

$$s_{semantic}(y, x) = s_{word}(y, x)^\beta \cdot s_{sent}(y, x)^\gamma \quad (3)$$

where β and γ are weighting hyperparameters.

Task-specific scorers. We apply TGLS to two tasks: paraphrasing and text formalization.

For paraphrasing, the goal is to generate a semantically similar but lexically different sentence. Previous work [24] uses the BLEU score to penalize the n -gram overlapping between the output and input: $s_{paraphrase}(y, x) = (1 - \text{BLEU}(y, x))^\delta$, which is also adopted in our work. Here, δ is a weighting hyperparameter for the task-specific scorer.

For text formalization, the goal is to transform an informal sentence to the formal style [36], which is an example of text style transfer. We follow the setting of most text style-transfer work [14], where we assume the style labels are available, but no parallel supervision is given. We train a classifier that predicts the probability of the style, also based on the RoBERTa features. Then, the task-specific scorer becomes $s_{formality}(y) = (p(\text{formal} | \text{RoBERTa}([\text{BOS}], y)))^\delta$, where δ is the weighting hyperparameter for this task.

Proposal of local edits. At a step t of SA search, a new candidate y' is proposed from $y^{(t)}$ by local editing. SA randomly picks a position to edit, as well as one of the following edits: Replace, Insert, and Delete.

For Replace, the model suggests a candidate word at x_i based on the posterior distribution induced by $s(y|x)$. For efficiency concerns, previous work [29, 24] evaluates top- K candidate words, suggested by a forward and backward language model. In our work, we adopt RoBERTa to evaluate the

posterior probability of a word, where the word embedding layer of RoBERTa at this slot is randomly masked. The Insert edit also suggests a word from the posterior, predicting a word given the newly added [MASK] token and the context. This complies with RoBERTa’s pretraining criteria of masked language modeling and is able to suggest high-quality candidate words. The Delete operator simply removes the word at a chosen position.

In text formalization, we have local edits based on a set of rules, e.g., “*we are*” substituting “*we’re*”, which are retrieved from PPDB[31]. Previous sequence-to-sequence approaches on this task adopt manually designed rules as a preprocessing step [36]. Our unsupervised TGLS, on the other hand, can easily make use of the off-the-shelf resources, since it can filter out the noise by rejecting the bad candidates.

In short, the SA search component in our TGLS mainly follows [24], but we re-design the scoring functions and the proposals. The main focus of this paper is to couple search and learning, especially the methods of training a machine learning model that learns from the search results, as follows.

2.2 Word-Level Cross-Entropy (CE) Learning

As mentioned in Section 1, the local search algorithm is computationally inefficient during inference time, because it requires a few hundred steps of edits and re-evaluations for each sample.

Our intuition is to train a conditional generative model, GPT2, based on SA’s search results. Specifically, an input x and SA’s searched sequence $y^{(SA)}$ are concatenated with a special separating token [SEP] in between, and the GPT2 is trained with losses on the y -part. Therefore, the GPT2 would be able to generate a output sequence directly from $p(y|x)$ in an autoregressive way.

Given a source sequence x , the objective is the word-by-word cross-entropy (CE) loss, given by

$$J_{CE} = - \sum_{n=1}^N \sum_{v \in \mathcal{V}} y_{i,v}^{(SA)} \log p_{i,v}^{(GPT2)} \quad (4)$$

where $y_{i,v}^{(SA)}$ is a binary value, indicating whether the i th word is v or not in the SA’s output for this data sample, and $p_{i,v}^{(GPT2)} = \Pr[y_i = v | y_{<i}^{(SA)}, x]$, which is predicted by the GPT2.

The word-level CE learning in TGLS adopts standard teacher-forcing technique with SA’s output being the pseudo-reference, i.e., during training, the GPT2 model learns the probability $p_{i,v}^{(GPT2)}$ at step i , assuming all previous words are correctly predicted as $y_{<i}^{(SA)}$. Thus, word-by-word CE trains all predictions in the sequence simultaneously, and is able to quickly adapt a generic pretrained GPT2 to the text generation task at hand.

It is also noted that minimizing the cross-entropy loss (4) is equivalent to minimizing $KL(\hat{y}_i^{(SA)} \| p_i^{(GPT2)})$, i.e., the KL-divergence between $\hat{y}_i^{(SA)}$ and $p_i^{(GPT2)}$, if viewed as distributions over the vocabulary. Due to the asymmetry nature, minimizing the KL-term makes the second slot $p_i^{(GPT2)}$ more wide-spreading than the first slot $\hat{y}_i^{(SA)}$, illustrated in Figure 1b. This provides an explanation of why the CE-trained GPT2 could smooth out the noise of the stochastic SA search. As will be shown in experiments, training a GPT2 from SA’s output alone can outperform SA.

2.3 Sequence-Level Maximum-Margin (MM) Learning

Our next insight is to perform alternations between search and learning to bootstrap performance of TGLS. In the first stage, SA starts local search with the initial candidate being the input (i.e., $y^{(0)} = x$), because we have no other meaningful candidate output yet. Starting with x takes advantage of the resemblance between input and output. But if a higher-quality candidate is available, SA may perform better than from x .

Therefore, we propose to have another stage of learning-from-search alternations. SA starts from an initial candidate being GPT2’s output, i.e., $y^{(0)} = y^{(GPT2)}$, shown in Figure 1c. Then, GPT2 is further fine-tuned to learn from the newly searched result. For the learning method, we propose to employ sequence-level max-margin (MM) training, instead of CE training. Such alternation can be performed for multiple epochs for performance bootstrapping.

Algorithm 1: Training TGLS

Input: A non-parallel corpus X **Output:** A fine-tuned GPT2 model

▷ First-stage search and learning

for an input $x \in X$ **do** $y^{(SA)} = SA(x, x)$ ▷ SA is detailed in Algorithm 2. In the first stage, SA starts with input x as the initial candidate**for all epochs do** **for an input** x **with its SA output** $y^{(SA)}$ **do** Fine-tune GPT2 by cross-entropy loss (4) with pseudo-reference $y^{(SA)}$, conditioned on x

▷ Second-stage search and learning

for all epochs do **for an input** x **do** $Y^{(GPT2)} = \text{BeamSearch}(\text{GPT2}(x))$ ▷ $Y^{(GPT2)}$ is a set of output by beam search $y^{(SA-S2)} = SA(x, y^{(GPT2)})$ for some $y^{(GPT2)} \in Y^{(GPT2)}$

▷ In the second stage, SA starts with GPT2’s output (any output in the beam is fine)

 $\tilde{Y} = Y^{(GPT2)} \cup \{y^{(SA-S2)}\}$

Fine-tune GPT2 with max-margin loss (5) with

 positive sample: $y^+ = \arg\max_{y \in \tilde{Y}} s(y|x)$, and negative samples: $\tilde{Y} \setminus \{y^+\}$ **Return:** Resulting GPT2 (denoted by GPT2-S2 after two stages of search and learning)

Concretely, the GPT2 trained with CE learning performs beam search (beam size B) and obtain a set of output sequences $Y^{(GPT2)} = \{y^{(GPT2,1)}, \dots, y^{(GPT2,B)}\}$. A randomly picked (for efficiency purpose) output in $Y^{(GPT2)}$ is taken as initial candidate in SA search, yielding a new sample $y^{(SA-S2)}$. We consider the set $\tilde{Y} = Y^{(GPT2)} \cup \{y^{(SA-S2)}\}$ as the positive and negative samples for MM learning. In fact, the positive sample y^+ is the best sequence scored by (1), i.e., $y^+ = \arg\max_{y \in \tilde{Y}} s(y|x)$. In most cases, we have $y^+ = y^{(SA-S2)}$, but this is not necessarily true because SA is not greedy. All other sentences in \tilde{Y} are collected as negative samples. We use the summation of GPT2’s pre-softmax logit as the negative energy.² In other words, we have $-E(y) = \sum_{i=1}^N z_{i,y_i}$ of a sequence $y = (y_1, \dots, y_N)$, where z_{i,y_i} is the logit for the word y_i at the i th step. The max-margin loss for this data sample is

$$J_{MM} = \sum_{y^- \in \tilde{Y}, y^- \neq y^+} \max \{0, -E(y^+) + E(y^-) - \Delta\} \quad (5)$$

where Δ is the margin hyperparameter.

In fact, the energy implicitly defines a globally normalized distribution as $p(y) = \frac{1}{Z} \exp\{-E(y)\}$, where Z is the partition function. The MM training increases the probability of the positive sample, while decreasing the probability of negative ones. In our MM training, the negative samples are given by beam search on GPT2, highly resembling the positive one. This makes TGLS more sensitive to the sequence-level scorer (1) in its probable region of the output space, illustrated in Figure 1d.

By contrast, word-level CE increases the probability of the target (analogous to the positive sample) step-by-step, while decreasing the probability of other samples due to local normalization. Thus, it cannot explicitly correct the prediction of a highly-probable but low-scored sample, and performs worse than MM in the second stage.

In summary, the training of TGLS involves two stages of search and learning, where CE and MM are used as the learning objective in different stages. Notice that, for the second stage, search and learning are alternated within the epoch loop. Thus, another stage of search and learning is unnecessary, because our second stage already allows multiple epochs for performance bootstrapping. For inference, we do not perform SA search, but directly use the fine-tuned GPT2 for autoregressive prediction.

²Energy is what MM learning would like to minimize for positive samples.

Appendix B further provides a detailed diagram of our TGLS.

2.4 Discussion: TGLS vs. Reinforcement Learning and Learning-to-Search

One of the most popular algorithms of reinforcement learning (RL) in text generation is the REINFORCE, which maximizes the expected reward (such as the BLEU score [35] or an adversarial discriminator [51]) by sampling a sequence of actions from its learned policy and reweighing the likelihood training of the sampled sequence. REINFORCE is known to have high variance, and previous REINFORCE-based text generation involves groundtruth pretraining [51]. Without a warm start, the sampling-based REINFORCE does not work with such a large action space as the vocabulary. Our TGLS would also optimize an external scoring function (analogous to the reward in RL), but does not have groundtruth for pretraining. We instead perform SA search and learn from SA’s (local) optima step-by-step.

Monte-Carlo Tree Search (MCTS) [40] is another paradigm of search and learning, where a search tree is maintained with each non-leaf node being a partial configuration (e.g., a partial sentence in text generation). Again, it suffers from the large branching factor, which is the vocabulary size in our applications. Our TGLS adopts local search, which maintains a full configuration and evaluates the candidate at each search step. The resemblance between input and output also largely eases the search task.

The Learning-to-Search (L2S) framework has been successfully applied to various NLP applications, such as structured prediction [5, 17] and text generation [46, 52]. L2S allows the model to explore/search in the space, collects the score (cost) for possible actions, and optimizes the model. Usually, L2S assumes an expert demonstration (groundtruth sequence and/or dynamic oracle) available as a reference policy L2S. For instance, a LaSO-like algorithm forces the model to search towards the groundtruth sequence; when the groundtruth is out of the search range, a learning update is performed, where the search effort serves as the negative samples and the groundtruth as positive examples for learning [6, 46]. By contrast, TGLS does not have groundtruth, but uses a strong search algorithm to find higher-scored sentences, which serve as positive samples.

3 Experiments

3.1 Datasets and Settings

Paraphrase Generation. Paraphrase generation is to rephrase input text with different expressions, while keeping the semantics. Following previous work [8, 12], we conducted experiments on the Quora benchmark dataset.³ As unsupervised text generation, we followed [24] and used 500K sentences to fine-tune GPT2 and RoBERTa for fluency and semantic scorers. For validation and testing, we had 500 and 170K samples, respectively.

We adopt BLEU and iBLEU as evaluation metrics, which are widely used for paraphrase generation. BLEU measures the length-penalized n -gram overlap between an output and the reference. In addition, paraphrasing requires that the output should be different from input. Thus, iBLEU [41] penalizes BLEU by n -gram similarity between output and input. Following most work, we consider iBLEU as the main metric for paraphrasing.

Text Formalization. This task concerns formality transfer of text, and our goal is to rephrase the given informal text into the formal style. We experimented with the Grammarly’s Yahoo Answers Formality Corpus (GYAFC) [36] in the domain of Family & Relationships. It is noted that GYAFC contains 50K informal-formal pairs, but our TGLS follows the setting of most other style-transfer work [14], which uses non-parallel corpora with style labels, but does not have parallel supervision. Our pretrained language models are additionally fine-tuned on automatically labeled non-parallel corpus [48]. In GYAFC, there are 3K samples for validation and 1K for test.

The performance of formality transfer is measured in different aspects. The language modeling perplexity evaluates the fluency of the generated text, and a learnable classifier predicts the formality accuracy. Particularly, the formality evaluator achieves an accuracy of 94%, being a good automatic

³<https://www.kaggle.com/c/quora-question-pairs>

Table 1: Automatic evaluation results on paraphrasing.

| Methods | iBLEU | BLEU |
|---|--------------|--------------|
| Supervised | | |
| RL-NN [33] | 14.83 | 20.98 |
| DAGGER [†] [7] | 18.88 | 28.42 |
| GPT2 [†] [34] | 19.19 | 26.92 |
| Distant supervised | | |
| Round-Trip MT (GPT2) [†] [11] | 11.24 | 16.33 |
| Round-Trip MT (Transformer) [†] [27] | 14.36 | 20.85 |
| Unsupervised | | |
| VAE [3] | 8.16 | 13.96 |
| CGMH [29] | 9.94 | 15.73 |
| UPSA [24] | 12.02 | 18.18 |
| SA w/ PLM (Ours) [†] | 14.52 | 21.08 |
| TGLS (Ours) [†] | 17.48 | 25.00 |

Table 2: Automatic evaluation results on formality transfer.

[‡]The smaller, the better.

| Methods [†] | PPL [‡] | BLEU | Formality | H-mean | G-mean |
|----------------------|------------------|--------------|--------------|--------------|--------------|
| Supervised | | | | | |
| LSTM-attn [36] | 23.42 | 69.36 | 87.39 | 77.34 | 77.85 |
| Unsupervised | | | | | |
| BackTrans [32] | 183.7 | 1.23 | 31.18 | 2.37 | 6.13 |
| StyleEmb [9] | 114.6 | 8.14 | 12.31 | 9.80 | 10.01 |
| MultiDec [9] | 187.2 | 13.29 | 8.18 | 10.13 | 10.42 |
| CrossAlign [39] | 44.78 | 3.34 | 67.34 | 6.36 | 14.99 |
| DelRetrGen [22] | 88.52 | 24.95 | 56.96 | 34.70 | 37.69 |
| Template [22] | 197.5 | 43.45 | 37.09 | 40.02 | 40.14 |
| UnsupMT [54] | 55.16 | 39.28 | 66.29 | 49.33 | 51.02 |
| DualRL [26] | 66.96 | 54.18 | 58.26 | 56.15 | 56.18 |
| TGLS (Ours) | 30.26 | 60.25 | 75.15 | 66.88 | 67.29 |

[†] indicates that the results are directly comparable to TGLS on the same data split. Appendix C provides more details on the baseline models and how these results are obtained.

evaluation measure.⁴ The BLEU score is also computed against the reference to evaluate n -gram overlap. Finally, we consider the harmonic mean (H-mean) and the geometric mean (G-mean) of the formality accuracy and the BLEU score as our main metrics for this task.

Hyperparameters. For SA, the initial temperature was set to 1e-2 in both tasks. The total search steps and temperature cooling were 50, 2e-4 for paraphrasing; and 100 and 1e-4 for text simplification. The scorers’ weights were tuned by grid search, set as $(\alpha, \beta, \gamma, \delta) = (0.8, 1, 0.6, 0.125)$ for paraphrasing, and $(0.8, 2, 1.25, 0.26)$ for text formalization. We keep the RoBERTa fixed and further tune the GPT2 model by alternations of searching-and-learning for another 6 epochs.

3.2 Overall Performance

Table 1 presents the results of automatic evaluation for paraphrase generation. Among the unsupervised approaches, the simulated annealing model UPSA [24] achieves the previous state-of-the-art performance, outperforming both variational sampling [3] and discrete-space Metropolis–Hastings sampling [29]. We propose to use large-scale pretrained language models for fluency and evaluation (mode name: SA w/ PLM), and improve iBLEU by 2.5 points from UPSA. Our TGLS framework of search and learning further improves iBLEU by 2.96 points, being a new state-of-the-art unsupervised paraphrasing model.

The TGLS also outperforms the paraphrasing systems based on round-trip translation, which is widely used in real-world applications. Such methods generate a paraphrase by translating a sentence to a foreign language and translating it back. It is categorized as distant supervision, because it requires parallel corpora for machine translation, but not for the paraphrasing task of interest.

Noticeably, our unsupervised TGLS performs comparably to a few recent paraphrasing model [33, 7]. Moreover, we train a GPT2 in the supervised setting for a controlled experiment, where the neural architecture is fixed. We see that the unsupervised TGLS is slightly worse than the supervised setting by only 1.71 iBLEU, largely closing the gap between supervised and unsupervised paraphrasing.

Table 2 presents the results for formality transfer. Again, we see consistent evidence on the effectiveness of TGLS, as it outperforms existing unsupervised approaches including heuristic marking of style words and retrieval-based editing [22], unsupervised machine translation approaches [54], and dual reinforcement learning [26].

Admittedly, the unsupervised TGLS is still worse than supervised approaches on this task. This is probably because our heuristic scorers are mostly designed for the paraphrasing task, and even for large-scale pretrained models, their performance may drop with informal text. More effort could be made here for future work.

⁴We reuse the architecture of RoBERTa for formality evaluation and GPT2 for fluency evaluation. However, they are separately trained, third-party models, and are NOT part of our TGLS.

Table 3: Model analysis on paraphrase generation. All variants use pretrained language models.

| Methods | iBLEU | BLEU | Inference Time (sec/sample) |
|--------------------|--------------|--------------|--------------------------------|
| SA | 14.52 | 21.08 | 5.46 |
| SA+CE | 14.97 | 23.25 | 0.06 |
| SA+CE+SA | 15.41 | 21.48 | 2.62 |
| SA+CE+SA+CE | 15.70 | 21.70 | 0.37 |
| SA+CE+SA+MM (full) | 17.48 | 25.00 | 0.43 |

We also conducted human evaluation, reported in Appendix D. Results are consistent with these automatic metrics.

3.3 Analysis

In this part, we present an in-depth analysis of our model with paraphrase generation as the testbed.

Ablation study. As TGLS involves two stages of search and learning, we conduct an ablation study, shown in Table 3. We start from a base simulated annealing (SA) approach, where we have already adopted pretrained language models. Thus, it sets up a fair comparison.

In the first stage of learning, our GPT2 model with word-level cross-entropy (CE) training already outperforms SA alone. The result is slightly surprising, but it actually makes sense because cross-entropy loss can smooth out the noise in SA’s heuristically defined search objective.

We also tried to train the GPT2 by max-margin (MM) loss without CE learning, but it fails to escape from a random policy. It is due to the difficulty of training an energy-based model in comparison to a locally normalized model [10]. In our work, the negative samples in the beam would be useless when the model is not warm started.

We compare SA with the initial sentence being input and GPT2’s prediction (SA vs. SA+CE+SA). We see the latter outperforms both SA and SA+CE. This confirms that the learned GPT2 helps SA find a better optimum.

The last two lines of Table 3 provide evidence of performance bootstrap by alternating between search and learning, as they outperform other ablated variants. In particular, MM is better than CE by a significant margin in the second stage. Our intuition is that MM with negative samples in the beam makes TGLS more sensitive in distinguishing sentence quality with its highly probable output region.

Inference efficiency. We also report computational time in Table 3. The experiments were conducted on a cluster with Nvidia Tesla V100 GPUs. The inference time could be noisy due to the multi-thread nature of clusters, but it provides a conclusive enough comparison between search-based and autoregressive generation. As seen, SA is inefficient because it requires hundreds of steps of editing and reevaluation. SA+CE, SA+CE+SA+CE, and SA+CE+SA+CE are all based on the GPT2 model during inference, and thus are much more computationally efficient. Based on the validation, SA+CE adopts greedy decoding, whereas the other two adopt beam search with a size of 5. We see all GPT2-based generators are at least 6–10× faster than the search-based methods.

Case Study. We present a case study in Appendix E. Typically examples show that TGLS is able to generate more fluent and more different-appearing paraphrases than search-based methods.

4 Related Work

Unsupervised text generation. There has been extensive work on neural unsupervised text generation. One popular approach is the variational auto-encoder [15], which is able to generate text by manipulated latent space for certain attributes, such as sentiment [14], topic [44], and syntax [53].

Recently, search-based methods has been developed for various text generation tasks, including text simplification [18], summarization[38], keyword-to-text generation [29], and paraphrasing [29, 24]. However, these methods are not learnable; hence, the inference is inefficient and the performance cannot be improved by training.

Most of other work of unsupervised text generation is built upon heuristics of certain tasks. For instance, Narayan and Gardent [30] propose a task-specific pipeline for sentence simplification. Zheng and Lapata [56] employ a graph-based ranking algorithm to select the most significant sentence as the summarization of a document. Chu and Liu [4] utilize the overlapping of text as a hint for multi-document summarization. In our work, the proposed TGLS is a generic framework, which also allows encoding prior knowledge of a task into the search algorithm.

Paraphrase generation. Most of the recent work on paraphrase generation focuses on neural models trained with large scale parallel-datasets. Several of them adopt different search techniques in supervised learning. The RL-based approaches propose to learn reward functions to score sampled sequence [23, 33, 50]. Besides, Du and Ji [7] finds that the learning-to-search approach can improve the performance of neural paraphrasing model. While our work mainly considers search and learning for unsupervised generation.

There is a group of work that utilizes (statistical- or neural-) machine translation (MT) systems, generating paraphrase by translating source sentences into a pivot language, and then translate it back into the original language [55, 27, 11]. Although no supervision of paraphrase is needed, the success of this approach depends on one or more high-quality MT systems, and hence large-scale parallel datasets for training translation models.

For unsupervised paraphrasing, one direction is to perform sampling by either a latent-space variational posterior sampler [2] or a word-space Metropolis–Hastings (MH) sampler [29]. By decreasing the temperature of the stationary distribution, Liu et al. [24] show that search-based formulation outperforms sampling for unsupervised text generation. Our work further extends it to a learning-from-search framework, improving both accuracy and inference efficiency.

Text style transfer. Our text formalization is one application of text style transfer. Other examples include sentiment [14] and the prose style [49]. Typically, text style transfer can be divided into three categories: parallel supervised, non-parallel supervised (with only style labels), and purely unsupervised. Parallel supervised style transfer trains a sequence-to-sequence model [36], whereas purely unsupervised style transfer relies on disentangling latent space [47].

Most previous work on text style transfer is in the non-parallel supervised setting, assuming style labels are available. Researchers have developed style embedding-based approaches [39, 9], style-specific decoders [9], style-word editing approaches [22], among others. Our approach also follows this setting, but to the best of our knowledge, we are the first to model style transfer as a search problem, as well as to extend it to the proposed TGLS framework of search and learning.

5 Conclusion

This work proposes TGLS, a novel framework of learning-from-search to unsupervised text generation. We show that the simulated annealing search can provide high-quality examples for the conditional text generator to learn from. Further, the improved generative model can give a better initial state to the search algorithm. Experiments demonstrate that the alternation of search and learning can boost the performance of TGLS on two unsupervised text generation tasks, paraphrase generation and text formalization. Moreover, our model is considerably more computationally efficient, compared with search-based generation methods. We note that TGLS opens a few future directions, such as more effective and efficient search algorithms, more noise-robust learning methods, and a better combination of search and learning. We would also like to apply the learning-from-search framework to other sequential prediction tasks in NLP.

References

- [1] R. K. Amplayo and M. Lapata. Unsupervised opinion summarization with noising and denoising. In *ACL*, 2020.
- [2] Y. Bao, H. Zhou, S. Huang, L. Li, L. Mou, O. Vechtomova, X. Dai, and J. Chen. Generating sentences from disentangled syntactic and semantic spaces. In *ACL*, pages 6008–6019, 2019.
- [3] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *CoNLL*, pages 10–21, 2015.
- [4] E. Chu and P. J. Liu. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *ICML*, pages 1223–1232, 2018.

- [5] H. Daumé III and D. Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*, pages 169–176, 2005.
- [6] H. Daumé III and D. Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*, pages 169–176, 2005.
- [7] W. Du and Y. Ji. An empirical comparison on imitation learning and reinforcement learning for paraphrase generation. In *EMNLP-IJCNLP*, pages 6014–6020, 2019.
- [8] Y. Fu, Y. Feng, and J. P. Cunningham. Paraphrase generation with latent bag of words. In *NeurIPS*, pages 13623–13634, 2019.
- [9] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan. Style transfer in text: Exploration and evaluation. In *AAAI*, pages 663–670, 2018.
- [10] K. Goyal, C. Dyer, and T. Berg-Kirkpatrick. An empirical investigation of global and local normalization for recurrent neural sequence models using a continuous relaxation to beam search. In *NAACL-HLT*, pages 1724–1733, 2019.
- [11] Y. Guo, Y. Liao, X. Jiang, Q. Zhang, Y. Zhang, and Q. Liu. Zero-shot paraphrase generation with multilingual language models. *arXiv preprint arXiv:1911.03597*, 2019.
- [12] A. Gupta, A. Agarwal, P. Singh, and P. Rai. A deep generative framework for paraphrase generation. In *AAAI*, pages 5149–5156, 2018.
- [13] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.
- [14] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward controlled generation of text. In *ICML*, pages 1587–1596, 2017.
- [15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [17] A. Krishnamurthy, H. D. CMU EDU III, and U. EDU. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*, 2015.
- [18] D. Kumar, L. Mou, L. Golab, and O. Vechtomova. Iterative edit-based unsupervised sentence simplification. In *ACL*, 2020.
- [19] G. Lample and A. Conneau. Cross-lingual language model pretraining. In *NeurIPS*, pages 7057–7067, 2019.
- [20] G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. In *ICLR*, 2018.
- [21] G. Lample, M. Ott, A. Conneau, L. Denoyer, and M. Ranzato. Phrase-based & neural unsupervised machine translation. In *EMNLP*, pages 5039–5049, 2018.
- [22] J. Li, R. Jia, H. He, and P. Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL-HLT*, pages 1865–1874, 2018.
- [23] Z. Li, X. Jiang, L. Shang, and H. Li. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*, 2017.
- [24] X. Liu, L. Mou, F. Meng, H. Zhou, J. Zhou, and S. Song. Unsupervised paraphrasing by simulated annealing. In *ACL*, 2020.
- [25] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [26] F. Luo, P. Li, J. Zhou, P. Yang, B. Chang, Z. Sui, and X. Sun. A dual reinforcement learning framework for unsupervised text style transfer. In *IJCAI*, pages 5116–5122, 2019.
- [27] J. Mallinson, R. Sennrich, and M. Lapata. Paraphrasing revisited with neural machine translation. In *ACL*, pages 881–893, 2017.
- [28] S. W. McRoy, S. Channarukul, and S. S. Ali. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420, 2003.

- [29] N. Miao, H. Zhou, L. Mou, R. Yan, and L. Li. CGMH: Constrained sentence generation by metropolis-hastings sampling. In *AAAI*, pages 6834–6842, 2019.
- [30] S. Narayan and C. Gardent. Unsupervised sentence simplification using deep semantics. In *INLG*, pages 111–120, 2015.
- [31] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*, pages 425–430, 2015.
- [32] S. Prabhume, Y. Tsvetkov, R. Salakhutdinov, and A. W. Black. Style transfer through back-translation. In *ACL*, pages 866–876, 2018.
- [33] L. Qian, L. Qiu, W. Zhang, X. Jiang, and Y. Yu. Exploring diverse expressions for paraphrase generation. In *EMNLP-IJCNLP*, pages 3164–3173, 2019.
- [34] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [35] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2017.
- [36] S. Rao and J. R. Tetreault. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*, pages 129–140, 2018.
- [37] S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, 1:1–20, 2010.
- [38] R. Schumann, L. Mou, Y. Lu, O. Vechtomova, and K. Markert. Discrete optimization for unsupervised sentence summarization with word-level extraction. In *ACL*, 2020.
- [39] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6830–6841, 2017.
- [40] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [41] H. Sun and M. Zhou. Joint learning of a dual SMT system for paraphrase generation. In *ACL*, volume 2, pages 38–42, 2012.
- [42] S. Surya, A. Mishra, A. Laha, P. Jain, and K. Sankaranarayanan. Unsupervised neural text simplification. In *ACL*, pages 2058–2068, 2019.
- [43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [44] W. Wang, Z. Gan, H. Xu, R. Zhang, G. Wang, D. Shen, C. Chen, and L. Carin. Topic-guided variational auto-encoder for text generation. In *NAACL*, pages 166–177, 2019.
- [45] J. Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
- [46] S. Wiseman and A. M. Rush. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, pages 1296–1306, 2016.
- [47] P. Xu, J. C. K. Cheung, and Y. Cao. On variational learning of controllable representations for text without supervision. In *ICML*, 2020.
- [48] R. Xu, T. Ge, and F. Wei. Formality style transfer with hybrid textual annotations. *arXiv preprint arXiv 1903.06353*, 2019.
- [49] W. Xu, A. Ritter, B. Dolan, R. Grishman, and C. Cherry. Paraphrasing for style. In *COLING*, pages 2899–2914, 2012.
- [50] Q. Yang, D. Shen, Y. Cheng, W. Wang, G. Wang, L. Carin, et al. An end-to-end generative architecture for paraphrase generation. In *EMNLP-IJCNLP*, pages 3123–3133, 2019.
- [51] L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.
- [52] W. Zhang, Y. Feng, F. Meng, D. You, and Q. Liu. Bridging the gap between training and inference for neural machine translation. In *ACL*, pages 4334–4343, 2019.

- [53] X. Zhang, Y. Yang, S. Yuan, D. Shen, and L. Carin. Syntax-infused variational autoencoder for text generation. In *ACL*, pages 2069–2078, 2019.
- [54] Z. Zhang, S. Ren, S. Liu, J. Wang, P. Chen, M. Li, M. Zhou, and E. Chen. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*, 2018.
- [55] S. Zhao, H. Wang, X. Lan, and T. Liu. Leveraging multiple mt engines for paraphrase generation. In *COLING*, pages 1326–1334, 2010.
- [56] H. Zheng and M. Lapata. Sentence centrality revisited for unsupervised summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6236–6247, 2019.

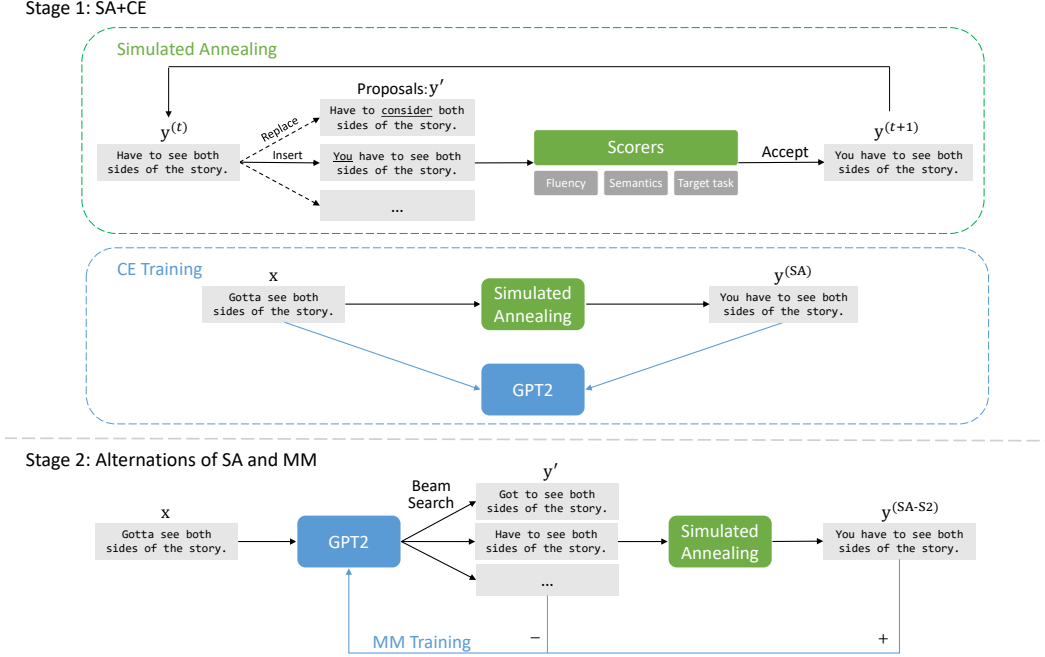


Figure 2: Two stages of search and learning in TGLS.

A Fine-Tuning Language Models for Scorers and Edit Proposal

Firstly, we describe the strategy of fine-tuning RoBERTa, which are used for scoring semantic compliance, and proposal of replacing words, in simulated annealing (Section 2.1). In particular, we consider two fine-tuning objectives, as follows.

Masked language model. This fine-tuning objective is based on domain-specific unlabeled corpora, and its goal is to adapt RoBERTa more specific to the domain at hand. For each experiment, we use its unlabeled training corpus for fine-tuning.

Generally, we follow the mixed masking strategy [25], which randomly masks out a few words a sentence. The goal is to predict these masked words. The mixed masking strategy randomly picks one of the three types of masking: (1) with probability 80%, the input is substituted by a special token [MASK]; (2) with probability 10%, the input is substituted by a random word in the vocabulary; and (3) with probability 10%, the input is substituted by the word itself (i.e., no masking is performed).

We observe that the first mask type aligns with the Replace and Insert proposals of local editing. Thus, we would high weigh this mask type in our fine-tuning. Each time we process a data sample, we perform one more masking with the special token [MASK], in addition to the mixed strategy.

Formality Classification. This objective is specific to text formalization experiment, where RoBERTa is also used for training a formality classifier. The objective is cross-entropy loss between $p(\text{formal} | \text{RoBERTa}(\text{BOS}, x))$ and the groundtruth formality label (formal or informal), where $\text{RoBERTa}(\text{BOS}, x)$ is the RoBERTa feature of a sentence x in the unlabeled dataset. Still, no parallel corpus is used. This fine-tuning objective works together with the masked language modeling objective in a multi-task fashion.

Note that the formality classification objective does not apply to the paraphrasing task.

As for GPT2, We first use the same strategy as pre-training to fine-tune it before search and learning. For paraphrasing, we fine-tune the model on all of the training set. For style transfer, we use the training set of formal style only.

Algorithm 2: SA for Text Generation [24]

Input: An input sentence x ,

An initial candidate output $y^{(0)}$

Output: An output sentence y

for $t = 1, \dots, \text{MaxStep}$ **do**

Set temperature $T = \max\{T_{\text{init}} - C \cdot t, 0\} \triangleright T_{\text{init}}, C$: annealing hyperparameters

Randomly pick an edit operator $\text{Op} \in \{\text{Delete}, \text{Insert}, \text{Replace}\}$

Randomly pick an edit position k

Propose a new candidate $y' = \text{Op}(y^{(t-1)}, k)$

Compute acceptance rate $p(\text{accept}|y', y^{(t-1)}, x, T) = \min\{1, \exp(\frac{s(y'|x) - s(y^{(t-1)}|x)}{T})\}$

$y^{(t)} = \begin{cases} y', & \text{with probability } p(\text{accept}|y', y^{(t-1)}, x, T) \\ y^{(t-1)}, & \text{with probability } 1 - p(\text{accept}|y', y^{(t-1)}, x, T) \end{cases}$

Return: $y^{(\text{SA})} = \text{argmax}_t s(y^{(t)}|x)$

For hyperparameters of language models fine-tuning, we performed 3 epochs of fine-tuning for text formalization and 9 epochs for paraphrasing. The maximum length of input was set to 35. We use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for optimization.

B Diagram of TGLS

Figure 2 shows the diagram of TGLS. Algorithm 1 further presents the pseudo-code of SA search [24] for reference.

C Competing Models

We present more details on the competing methods in Tables 1 and 2. All metrics are computed based on word-level tokenization (i.e., no BPE segmentation is used).

Paraphrase Generation

- **RL-NN.** Qian et al. [33] propose to learn a reward function by neural networks and perform REINFORCE training. The results in Table 1 are from the original paper.
- **DAGGER.**[†] Du et al. [7] apply imitation learning to paraphrase generation. It achieves the state-of-the-art performance on Quora dataset. We re-ran their implementation based on our data split.
- **GPT2.**[†] We train another GPT2 with the same hyperparameters as our TGLS, but in a supervised setting for a controlled comparison.
- **Round-Trip MT (Transformer).** Following [27], we utilize a well-trained bi-directional neural machine translation (NMT) system (Zh→En and En→Zh) with a Transformer model [43]. The NMT system achieves BLEU scores of 43.2 (En→Zh) and 28.74 (Zh→En) on the Newstest 2017 dataset. In our work, we use the roundtrip-translation (En→Zh→En) as the paraphrase.
- **Round-Trip MT (GPT2).** Similarly, we adopt another GPT2-based multi-lingual (En, Zh, Es, Ru) NMT system in [11]. Suggested by [11], we take the Zh as the pivot language.
- **VAE.** Variational autoencoders [3] generates a paraphrase by sampling from the encoded posterior distribution in the latent space. Here, we quote the results of CGMH from the implementation of [24].
- **CGMH.** Miao et al. [29] propose a word-space Metropolis–Hastings approach to paraphrase generation. Results also quoted from the implementation of [24].
- **UPSA.** Liu et al. [24] extend CGMH by decreasing the temperature and this becomes simulated annealing. Results are quoted from the original paper.

Table 4: Human evaluation on Quora dataset

| Method | Grammar, Fluency | Coherency, Consistency | Agreement |
|-----------|---------------------|---------------------------|-----------|
| UPSA [24] | 4.05 | 3.28 | 35.0% |
| SA w/ PLM | 4.79 | 4.48 | 70.0% |
| Our TGLS | 4.85 | 4.66 | 78.8% |

- **SA w/ PLM.**[†] One of our extensions to UPSA is to fine-tune pretrained language models for the search objective and edit proposals. This variant is essentially the intermediate results of our TGLS, after its first-stage SA search.

While widely used for paraphrasing, the Quora dataset does not contain a standard split. The dataset is crawled from the Internet, and thus it is noisy and sometimes contains duplicate samples in training and test sets. This would not be a severe problem if the duplication is between training input and reference output during supervised learning; thus, most previous work does not explicitly deduplicate these samples. However, this could affect our TGLS, because we perform learning from search results with the non-parallel training set. Thus, we carefully handled this problem, ensuring no overlap in training and test.

The competing models with [†] indicate that the data split is the same as TGLS, and the results are directly comparable. Others can be compared in a statistical sense.

Text Formalization

- **LSTM-attn.** Rao et al. [36] trained a Bi-LSTM encoder-decoder model with attention on their parallel formality corpus.
- **BackTrans.** Prabhumoye et al [32] utilizes back-translation to get style-independent content representations and feed them to style-dependent decoder to control the style of output.
- **StyleEmb.** Fu et al. [9] propose two variants for style transfer. In this variant, they accomplish style transfer by a learned style embedding.
- **MultiDec.** The other variant of [9] use multiple decoders for style-specific generation.
- **CrossAlign.** Shen et al. [39] also use style embedding, but they apply adversarial training based on style-transferred hidden states to cross-align content.
- **DelRetrGen.** Li et al. [22] propose a heuristic-based approach to mark style-specific words and phrases, and obtain expressions in a desired style by retrieval. Eventually, a neural model generates a style-transferred sentence.
- **Template.** This is a simpler variant in [22]. Then the detected style-specific words of input sentences are replaced by stylized words of target domain within its retrieved counterpart.
- **UnsupMT.** Zhang et al. [54] apply unsupervised machine translation techniques for style transfer. They firstly conduct word-to-word transfer and construct pseudo sentence pairs for system initialization, then conduct iterative back-translation training.
- **DualRL.** Luo et al. [26] use a dual reinforcement learning strategy to learn bi-directional style transfer without explicitly separating the style and content.

The results in Table 2 involve learnable metrics. We used separately trained GPT2 and RoBERT for fluency and formality evaluation, respectively. The GYAFC corpus has a standard dataset split. For fairness, we re-evaluated all the outputs based on our own evaluation models.

The outputs of LSTM-attn are released by [36], and the rest outputs are published by [26].

D Human Evaluation

We conduct human evaluation for the paraphrase generation experiment with selected baselines that are most relevant to our work, due to the limit of budgets. We randomly selected 120 paraphrase samples. Four annotators with linguistic background were invited to assess each sentence pair on a scale of 1–5 with regard to two criteria: relevance (coherence & consistent) and naturalness (grammar

Table 5: Examples generated by SA (w/ PLM) and the full TGLS model.

| # | Source | SA w/ PLM | TGLS |
|---|--|--|--|
| 1 | how better is sony mdr-xb950ap than sony mdr-xb450? | how better is the new sony mdr-xb or the sony mdr-xb? | how good is the sony mdr-xb950 vs the sony mdr-xb450? |
| 2 | what is meant by british political conservatism? | what is meant by the british? | what is the british political conservatism? |
| 3 | what are the problems occur when flow velocity increases in a pipe? | what are the problems that arise when the velocity of an increase in a cylinder increases? | what are some problems when the flow velocity of a pipe increases? |
| 4 | if a person gets out of a vehicle can they still receive a citation? | can a person gets off of a vehicle, do they still receive a citation? | can a person get out of a car without a citation? |

& fluency) in a blind manner. Each example was reviewed by two judges and average scores are reported in Table 4. We also show the percentage of agreement between human annotators. As seen, humans have a higher agreement when the model performance is high; otherwise, the agreement is lower, because different annotators have their own scale among 1–5.

In terms of performance, our extension of UPSA—with pretrained language models for fluency and semantic scorers—indeed improves human satisfaction in terms of Grammar/Fluency and Coherence/Consistency. Our search-and-learning TGLS model further improves the performance in both aspects. The results are also consistent with the automatic metrics in Section 3.2.

E Case Study

We show in Table E examples generated by SA (with pretrained language models) and the full TGLS. As seen, SA sometimes does not generate fluent sentences. In Example 2, the phrase “*political conservative*” is deleted but no synonyms are suggested as a replacement. Our TGLS is able to generate more fluent sentences. Moreover, our TGLS generates a paraphrase in an autoregressive fashion, thus sometimes yielding a more different-appearing output, e.g., “*flow velocity increases in a pipe*” being rephrased to “*flow velocity of a pipe increases*” in Example 3.

In Example 4, we also see that TGLS generates a seemingly plausible paraphrase given the input. However, the output conveys an opposite intention to the input. This shows that understanding the logic and pragmatics of language is still difficult for large-scale pretrained language models, and deeper semantic analysis shall be addressed in future work.