Unsupervised Paraphrasing by Simulated Annealing

Xianggen Liu,[1] **Lili Mou,**[2] **Fandong Meng,**[3] **Hao Zhou,**[4] **Jie Zhou,**[3] **Sen Song**[1]
[1]Tsinghua University, [2]University of Alberta [3]WeChat AI, Tencent Inc,[4]ByteDance AI Lab
liuxg16@mails.tsinghua.edu.cn, doublepower.mou@gmail.com, fandongmeng@tencent.com,
zhouhao.nlp@bytedance.com, withtomzhou@tencent.com, songsen@tsinghua.edu.cn

Abstract

Unsupervised paraphrase generation is a promising and important research topic in natural language processing. We propose UPSA, a novel approach that accomplishes Unsupervised Paraphrasing by Simulated Annealing. We model paraphrase generation as an optimization problem and propose a sophisticated objective function, involving semantic similarity, expression diversity, and language fluency of paraphrases. Then, UPSA searches the sentence space towards this objective by performing a sequence of local editing. Our method is unsupervised and does not require parallel corpora for training, so it could be easily applied to different domains. We evaluate our approach on a variety of benchmark datasets, namely, Quora, Wikianswers, MSCOCO, and Twitter. Extensive results show that UPSA achieves the state-of-the-art performance compared with previous unsupervised methods in terms of both automatic and human evaluations. Further, our approach outperforms most existing domain-adapted supervised models, showing the generalizability of UPSA.[1]

Introduction

Paraphrasing aims to restate one sentence as another with the same meaning, but different wordings. It constitutes a corner stone in many NLP tasks, such as question answering (Mckeown 1983), information retrieval (Knight and Marcu 2000), and dialogue systems (Shah et al. 2018). However, automatically generating accurate and different-appearing paraphrases is a still challenging research problem, due to the complexity of natural language.

Conventional approaches (Prakash et al. 2016; Gupta et al. 2018) model the paraphrase generation as a supervised encoding-decoding problem, inspired by machine translation systems. Usually, such models require massive parallel samples for training. In machine translation, for example, the WMT 2014 English-German dataset contains 4.5M sentence pairs (Neidert et al. 2014).

However, the training corpora for paraphrasing are usually small. The widely-used Quora dataset[2] only contains 140K pairs of paraphrases; constructing such human-written paraphrase pairs is expensive and labor-intensive. Further, existing paraphrase datasets are domain-specific: the Quora dataset only contains question sentences, and thus, supervised paraphrase models do not generalize well to new domains (Li et al. 2019). On the other hand, researchers synthesize pseudo-paraphrase pairs by clustering news events (Barzilay and Lee 2003), crawling tweets of the same topic (Lan et al. 2017), or translating bi-lingual datasets (Wieting and Gimpel 2017), but these methods typically yield noisy training sets, leading to low paraphrasing performance (Li et al. 2018).

As a result, unsupervised methods would largely benefit the paraphrase generation task if no parallel data are needed. With the help of deep learning, researchers are able to generate paraphrases by sampling from a neural network-defined probabilistic distribution, either in a continuous latent space (Bowman et al. 2016) or directly in the word space (Miao et al. 2019). However, the meaning preservation and expression diversity of those generated paraphrases are less "controllable" in such probabilistic sampling procedures.

To this end, we propose a novel approach to *Unsupervised Paraphrasing by Simulated Annealing* (UPSA). Simulated annealing (SA) is a stochastic searching algorithm towards an objective function, which can be flexibly defined. In our work, we design a sophisticated objective function, considering semantic preservation, expression diversity, and language fluency of paraphrases. SA searches towards this objective by performing a sequence of local editing steps, namely, word replacement, insertion, and deletion. For each step, UPSA first proposes a potential editing, and then accepts or rejects the proposal based on sample quality. In general, a better sentence (higher scored in the objective) is always accepted, while a worse sentence is likely to be rejected, but could also be accepted (controlled by an annealing temperature). At the beginning, the temperature is usually high, and worse sentences are more likely to be accepted. This pushes the SA algorithm outside a local optimum. The temperature is cooled down as the optimization proceeds, making the model better settle down to some optimum. Figure 1 illustrates how UPSA searches an optimum

[1]Code and data are available at: https://github.com/anonymity-person/UPSA (anonymized during double-blind review).

[2]https://www.kaggle.com/c/quora-question-pairs

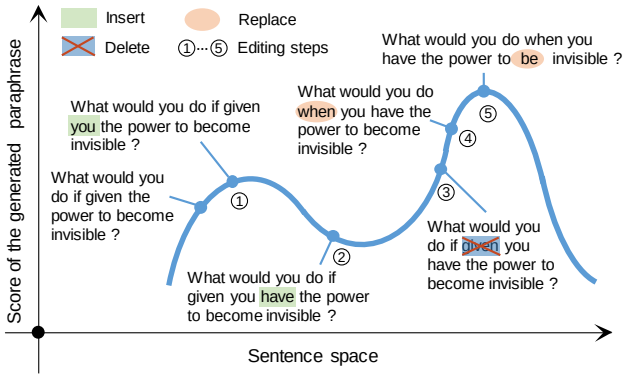

Figure 1: UPSA generates a paraphrase by a series of editing operations (i.e., insertion, replacement, and deletion). At each step, UPSA proposes a candidate modification of the sentence, which is accepted or rejected according to a certain acceptance rate (only accepted modifications are shown). Although sentences are discrete, we make an analogue in the continuous real x-axis where the distance of two sentences is roughly given by the number of edits.

in unsupervised paraphrase generation.

We evaluate the effectiveness of our model on four paraphrasing datasets, namely, Quora, Wikianswers, MSCOCO, and Twitter. Experimental results show that UPSA achieves a new state-of-the-art unsupervised performance in terms of both automatic metrics and human evaluation. Our unsupervised approach also outperforms most domain-adapted paraphrase generators.

In summary, our contributions are as follows:

- We propose the novel UPSA framework for Unsupervised Paraphrasing approach by Simulated Annealing.
- We design a searching objective function of SA that not only considers language fluency and semantic similarity, but also explicitly models expression diversity between a paraphrase and the input.
- We propose a copy mechanism as one of our searching action during simulated annealing to address rare words.
- We achieve the state-of-the-art performance on four benchmark datasets among all unsupervised paraphrase generators, largely reducing the performance gap between unsupervised and supervised paraphrasing. We outperform most domain-adapted paraphrase generators, and even a supervised one on the Wikianswers dataset.

Related Work

In early years, paraphrasing is typically accomplished by exploiting linguistic knowledge, including handcrafted rules (Mckeown 1983), grammar structures (Ellsworth and Janin 2007; Narayan, Reddy, and Cohen 2016), and shallow features (Zhao et al. 2009). Recently, deep neural networks have become a prevailing approach for text generation (Gupta et al. 2018), where paraphrasing is often formulated as a supervised encoding-decoding problem, for example, stacked residual LSTM (Prakash et al. 2016) and the Transformer model (Wang et al. 2019).

Li et al. (2019) learn paraphrasing at different levels of granularity (namely, sentence- and word-level paraphrasing), also in a supervised fashion. This achieves the state-of-the-art performance of paraphrase generation and is more generalizable to new domains.

Unsupervised paraphrasing is an emerging research direction in the field of NLP. The variational autoencoder (VAE) can be intuitively applied to paraphrase generation in an unsupervised fashion, as we can sample sentences from a learned latent space (Bowman et al. 2016). But the generated sentences are less controllable and suffer from the error accumulation problem in VAE's decoding phase (Miao et al. 2019).

Roy and Grangier (2019) introduce an unsupervised model based on vector-quantized autoencoders (Van den Oord, Vinyals, and others 2017). But their work mainly focuses on generating sentences for data argumentation instead of paraphrasing itself, and is not directly comparable.

Miao et al. (2019) use Metropolis–Hastings sampling (1953) for constrained sentence generation, achieving the state-of-the-art unsupervised paraphrasing performance. The main difference between their work and ours is that we formulate paraphrasing as a stochastic searching problem. In addition, we define our searching objective involving not only semantic similarity and language fluency, but also the expression diversity; we further propose a copy mechanism in our searching process.

Recently, a few studies have applied editing-based approaches to sentence generation. Guu et al. (2018) propose a heuristic delete-retrieve-generate approach as a component of a supervised sequence-to-sequence (Seq2Seq) model, but our UPSA is a mathematically inspired, unsupervised searching algorithm. Dong et al. (2019) learn the deletion and insertion operations for text simplification in a supervised way, where their groundtruth operations are obtained by some dynamic programming algorithm. Our editing operations (insertion, deletion, and replacement) are the searching actions of unsupervised simulated annealing.

Regarding discrete optimization/searching, a naïve approach is by hill climbing (Edelkamp and Schroedl 2011), which is in fact a greedy algorithm. In NLP, beam search (BS, Lowerre and Reddy 1980) is widely applied to sentence generation. BS maintains a k-best list in a partially greedy fashion during left-to-right (or right-to-left) decoding (Anderson et al. 2017). By contrast, UPSA makes distributed modifications over the entire sentence. Moreover, UPSA is able to make use of the original sentence as an initial state of searching, whereas BS usually works in the decoder of a sequence-to-sequence model and is not applicable to unsupervised paraphrasing.

Approach

In this section, we present our novel UPSA framework that uses simulated annealing (SA) for unsupervised paraphrase generation. In particular, we first present the general SA algorithm, and then design our searching objective and searching actions (i.e., candidate sentence generator).

The Simulated Annealing Algorithm

Simulated Annealing (SA) is an effective and general meta-heuristic of searching, especially for a large discrete or continuous space (Kirkpatrick, Gelatt, and Vecchi 1983).

Let $\mathcal{X}$ be a (huge) search space of sentences, and $f(\mathrm{x})$ be an objective function. The goal is to search for a sentence x that **maximizes** $f(\mathrm{x})$. At a searching step t, SA keeps a current sentence x_t, and proposes a new candidate x_* by local editing. If the new candidate is better scored by f, i.e., $f(\mathrm{x}_*) > f(\mathrm{x}_t)$, then SA accepts the proposal. Otherwise, SA tends to reject the proposal x_*, but may still accept it with a small probability $e^{\frac{f(\mathrm{x}_*)-f(\mathrm{x}_t)}{T}}$, controlled by an annealing temperature T. In other words, the probability of accepting the proposal is

$$p(\text{accept}|\mathrm{x}_*, \mathrm{x}_t, T) = \min\left(1, e^{\frac{f(\mathrm{x}_*)-f(\mathrm{x}_t)}{T}}\right). \quad (1)$$

If the proposal is accepted, $\mathrm{x}_{t+1} = \mathrm{x}_*$, or otherwise, $\mathrm{x}_{t+1} = \mathrm{x}_t$.

Inspired by the annealing in chemistry, the temperature T is usually high at the beginning of searching, leading to a high acceptance probability even if x_* is worse than x_t. Then, the temperature is decreased gradually as the search proceeds. In our work, we adopt the linear annealing schedule, given by $T = \max(0, T_{\text{init}} - C \cdot t)$, where T_{init} is the initial temperature and C is the decreasing rate.

The high initial temperature of SA makes the algorithm less greedy compared with hill climbing, whereas the decreasing of temperature along the search process enables it to better settle down to a certain optimum.

Theoretically, simulated annealing is guaranteed to converge to the global optimum in a finite problem if the proposal and the temperature satisfy some mild conditions (Granville, Krivanek, and Rasson 1994). Although such convergence may be slower than exhaustive search and the sentence space is, in fact, potentially infinite, simulated annealing is still a widely applied searching algorithm, especially for discrete optimization. Readers may refer to Hwang (1988) for details of the SA algorithm.

Objective Function

Simulated annealing maximizes an objective function, which can be flexibly specified in different applications. In particular, our UPSA objective $f(\mathrm{x})$ considers multiple aspects of a candidate paraphrase, including semantic preservation f_{sem}, expression diversity f_{exp}, and language fluency f_{flu}. Thus, our searching objective is to maximize

$$f(\mathrm{x}) = f_{\text{sem}}(\mathrm{x}, \mathrm{x}_0) \cdot f_{\text{exp}}(\mathrm{x}, \mathrm{x}_0) \cdot f_{\text{flu}}(\mathrm{x}), \quad (2)$$

where x_0 is the input sentence.

Semantic Preservation. A paraphrase is expected to capture all the key semantics of the original sentence. Thus, we leverage the cosine function of keyword embeddings to measure if the key focus of the candidate paraphrase is the same as the input. We use Rose et al. (2010) to extract the keywords of the input sentence x_0 and embed them by GloVE (Pennington, Socher, and Manning 2014). For each keyword, we find the closest word in the candidate paraphrase x_* in terms of cosine similarity. Our keyword-based semantic preservation score is given by the lowest cosine similarity among all keywords, i.e., the least matched keyword:

$$f_{\text{sem,key}}(\mathrm{x}_*, \mathrm{x}_0) = \min_{e \in \text{keywords}(\mathrm{x}_0)} \max_j \{\cos(\boldsymbol{w}_{*,j}, \boldsymbol{e})\}, \quad (3)$$

where $w_{*,j}$ is the jth word in the sentence x_*; e is an extracted keyword of x_0. Bold letters indicate embedding vectors.

In addition to keyword embeddings, we also adopt a sentence-level similarity function, based on Sent2Vec embeddings (Pagliardini, Gupta, and Jaggi 2017). Sent2Vec learns n-gram embeddings and computes the average of n-grams embeddings as the sentence vector. It has been shown significant improvements over other unsupervised sentence embedding methods in similarity evaluation tasks (Pagliardini, Gupta, and Jaggi 2017). Let $\mathbf{x}_*$ and $\mathbf{x}_0$ be the Sent2Vec embeddings of the candidate paraphrase and the input sentence, respectively. Our sentence-based semantic preservation scoring function is $f_{\text{sim,sen}}(\mathrm{x}_*, \mathrm{x}_0) = \cos(\mathbf{x}_*, \mathbf{x}_0)$.

To sum up, the overall semantic preservation scoring function of UPSA is given by

$$f_{\text{sem}}(\mathrm{x}_*, \mathrm{x}_0) = f_{\text{sem,key}}(\mathrm{x}_*, \mathrm{x}_0)^P \cdot f_{\text{sem,sen}}(\mathrm{x}_*, \mathrm{x}_0)^Q, \quad (4)$$

where P and Q are hyperparameters, balancing the importance of the two factors. Here, we use power weights because the scoring functions are multiplicative.

Expression Diversity. The expression diversity scoring function computes the lexical difference of two sentences. We adopt a BLEU-induced function to penalize the repetition of the words and phrases in the input sentence:

$$f_{\text{exp}}(\mathrm{x}_*, \mathrm{x}_0) = (1 - \text{BLEU}(\mathrm{x}_*, \mathrm{x}_0))^S, \quad (5)$$

where the BLEU score (Papineni et al. 2002) computes a length-penalized geometric mean of n-gram precision ($n = 1, \cdots, 4$). S coordinates the importance of $f_{\text{exp}}(\mathrm{x}_t, \mathrm{x}_0)$ in the objective function (2).

Language Fluency. Despite semantic preservation and expression diversity, the candidate paraphrase should be a fluent sentence by itself. We use a separately trained (forward) language model (denoted as $\overrightarrow{\text{LM}}$) to compute the likelihood of the candidate paraphrase as our fluency scoring function:

$$f_{\text{flu}}(\mathrm{x}_*) = \prod_{k=1}^{k=l_*} p_{\overrightarrow{\text{LM}}}(w_{*,k}|w_{*,1}, \ldots, w_{*,k-1}), \quad (6)$$

where l_* is the length of x_* and $w_{*,1}, \ldots, w_{*,l}$ are words of x_*. Here, we use a dataset-specific language model, trained on non-parallel sentences. Notice that a weighting hyperparameter is not needed for f_{flu}, because the relative weights of different factors in Eqn. (2) are given by the powers in $f_{\text{sem,key}}$, $f_{\text{sem,sen}}$, and f_{exp}.

Candidate Sentence Generator

As mentioned, simulated annealing proposes a candidate sentence at each searching action, which is either accepted or rejected by Eqn. (1). Since each action yields a new sentence x_* from x_t, we call it a *candidate sentence generator*.

While the proposal of candidate sentences does not affect convergence in theory (if some mild conditions are satisfied), it may largely influence the efficiency of SA searching.

In our work, we mostly adopt the word-level editing in Miao et al. (2019) as our searching actions. At each step t, the candidate sentence generator randomly samples an editing position k and an editing operation namely, replacement, insertion, and deletion. For replacement and insertion, the candidate sentence generator also samples a candidate word.

Let the current sentence be $\mathrm{x}_t = (w_{t,1}, \ldots, w_{t,k-1}, w_k, w_{t,k+1} \ldots, w_{t,l_t})$. If the replacement operation proposes a candidate word w_* for the kth step, the resulting candidate sentence becomes $\mathrm{x}_* = (w_{t,1}, \ldots, w_{t,k-1}, w_*, w_{t,k+1} \ldots, w_{t,l_t})$. The insertion operation works similarly.

Here, the word is sampled from a probabilistic distribution, induced by the objective function (2):

$$p(w_*|\cdot) = \frac{f_{\text{sim}}(\mathrm{x}_*, \mathrm{x}_0) \cdot f_{\text{exp}}(\mathrm{x}_*, \mathrm{x}_0) \cdot f_{\text{flu}}(\mathrm{x}_*)}{Z}, \quad (7)$$

$$Z = \sum_{w_* \in \mathcal{W}} f_{\text{sim}}(\mathrm{x}_*, \mathrm{x}_0) \cdot f_{\text{exp}}(\mathrm{x}_*, \mathrm{x}_0) \cdot f_{\text{flu}}(\mathrm{x}_*), \quad (8)$$

where $\mathcal{W}$ is the sampling vocabulary; Z is known as the normalizing factor (noticing our scoring functions are non-negative). We observe that sampling from such objective-induced distribution typically yields a meaningful candidate sentence, which enables SA to explore the search space more efficiently.

It is also noted that sampling a word from the entire vocabulary involves re-evaluating (2) for each candidate word, and therefore, we also follow Miao et al. (2019) and only sample from the top-K words given by jointly considering a forward language model and backward language model. The replacement operator, for example, suggests the top-K words vocabulary by

$$\mathcal{W}_{t,\text{replace}} = \text{top-}K_{w_*} \Big[p_{\overrightarrow{\text{LM}}}(w_{t,1}, \ldots, w_{t,k-1}, w_*) \cdot p_{\overleftarrow{\text{LM}}}(w_*, w_{t,k+1}, \ldots, w_{t,l_t}) \Big]. \quad (9)$$

For word insertion, the top-K vocabulary $\mathcal{W}_{t,\text{insert}}$ is computed in a similar way (except that the position of w_* is slightly different). Details are not repeated. In our experiments, K is set to 50.

Copy Mechanism. We observe that name entities and rare words are sometimes deleted or replaced during SA stochastic sampling. They are difficult to be recovered because they usually have a low language model-suggested probability.

Therefore, we propose a copy mechanism for SA sampling, inspired by that in Seq2Seq learning (Gu et al. 2016). Specifically, we allow the candidate sentence generator to copy the words from the original sentence x_0 for word replacement and insertion. This is essentially enlarging the top-K sampling vocabulary with the words in x_0, given by

$$\widetilde{\mathcal{W}}_{t,\text{op}} = \mathcal{W}_{t,\text{op}} \cup \{w_{0,1}, \ldots, w_{0,l_0}\}; \quad \text{op} \in \{\text{replace,insert}\} \quad (10)$$

Thus, $\widetilde{\mathcal{W}}_{t,\text{op}}$ is the actual vocabulary from which SA samples the word w_* for replacement and insertion operation.

Algorithm 1 UPSA

1: **Input**: Original sentence x_0
2: **for** $t \in \{1, \ldots, N\}$ **do**
3: $T = \max\{T_{\text{init}} - C \cdot t, 0\}$
4: Randomly choose an editing operation and a position k
5: Obtain a candidate x_* by candidate sentence generator
6: Compute the accepting probability p_{accept} by Eqn. (1)
8: With probability p_{accept}, $\mathrm{x}_{t+1} = \mathrm{x}_*$
9: With probability $1 - p_{\text{accept}}$, $\mathrm{x}_{t+1} = \mathrm{x}_t$
10: **end for**
10: **return** x_τ s.t. $\tau = \text{argmax}_{\tau \in \{1,\ldots,N\}} f(\mathrm{x}_\tau)$

While such vocabulary reduces the proposal space, it works well empirically because other low-ranked candidate words are either irrelevant or makes the sentence influent; they usually have low objective scores, and are likely to be rejected even if sampled.

Overall Optimization Process

We summarize our simulated annealing algorithm for unsupervised paraphrasing (UPSA), also shown in Algorithm 1.

Given an input x_0, UPSA searches from the sentence space to maximize our objective $f(\mathrm{x})$, which involves semantic preservation, expression diversity, and language fluency. UPSA starts from x_0 itself. For each searching step, it randomly selects a searching action (namely, word insertion, deletion, and replacement) at a position k (Line 4); if insertion or replacement is selected, UPSA also proposes a candidate word, so that a candidate paraphrase x_* is formed (Line 5). Then, UPSA computes an acceptance rate p_{accept} based on the increment of f and the temperature T (Line 6). The candidate sentence x_{t+1} for the next step becomes x_t if the proposal is accepted, or remains x_t if the proposal is rejected. Until the maximum searching iterations, we choose the sentence x_τ that yields the highest score.

Experiments

Datasets

Quora. The Quora question pair dataset (Footnote 2) contains 140K parallel sentences and additional 640K non-parallel sentences. We follow the unsupervised setting in Miao et al. (2019), where there are 3K and 20K pairs for validation and test, respectively.

Wikianswers. The original Wikianswers dataset (Fader, Zettlemoyer, and Etzioni 2013) contains 2.3M pairs of question paraphrases from the Wikianswers website.[3] Since our model only involves training a language model, we randomly selected 500K non-parallel sentences for training. For evaluation, we followed the same protocol as Li et al. (2019) and randomly sampled 5K for validation and 20K for testing. Although the exact data split in previous work is not available, our results are comparable to previous ones in the statistical sense.

MSCOCO. The MSCOCO dataset contains 500K+ paraphrases pairs for ∼120K image captions (Lin et al. 2014).

[3] http://knowitall.cs.washington.edu/paralex/

We follow the standard split (Lin et al. 2014) and the evaluation protocol in Prakash et al. (2016) where only image captions with fewer than 15 words are considered.

Twitter. The Twitter URL paraphrasing corpus (Lan et al. 2017) is originally constructed for paraphrase identification. We follow the standard train/test split, but take 10% of the training data as the validation set. The remaining samples are used to train our language model. For the test set, we only consider sentence pairs that are labeled as "paraphrases." This results in 566 test cases.

Competing Methods and Metrics

Unsupervised paraphrasing is an emerging research topic, and we could only find two plausible competing methods (namely, VAE and CGMH) in this setting. Early work on unsupervised paraphrasing typically adopts rule-based methods (Mckeown 1983; Barzilay and Lee 2003). Their performance could not be verified on the above datasets, since the extracted rules are not available. Therefore, we are unable to compare them in this paper. Also, rule-based systems usually do not generalize well to different domains. In the following, we describe our competing methods:

• **VAE.** We train a variational autoencoder (VAE) with two-layer, 300-dimensional LSTM units.[4] The VAE is trained with non-parallel corpora by maximizing the variational lower bound of log-likelihood; during inference, sentences are sampled from the learned variational latent space (Bowman et al. 2016).

• **CGMH.** Miao et al. (2019) use Metropolis–Hastings sampling in the word space for constrained sentence generation. It is shown to outperform latent space sampling as in VAE, and is the state-of-the-art unsupervised paraphrasing approach. We adopted the published source code and generated paraphrases for comparison.

We further compare UPSA with supervised Seq2Seq paraphrase generators: ResidualLSTM (Prakash et al. 2016), VAE-SVG-eq (Gupta et al. 2018), Pointer-generator (See, Liu, and Manning 2017), the Transformer (Vaswani et al. 2017), and the decomposable neural paraphrase generation (DNPG, Li et al. 2019). DNPG has been reported as the state-of-the-art supervised paraphrase generator.

To better compare UPSA with all paraphrasing settings, we also include domain-adapted supervised paraphrase generators that are trained in a source domain but tested in a target domain, including shallow fusion (Gulcehre et al. 2015) and multi-task learning (MTL, Domhan and Hieber 2017).

We adopt BLEU (Papineni et al. 2002) and ROUGE (Lin 2004) scores as automatic metrics to evaluate model performance. Sun and Zhou (2012) observe that BLEU and ROUGE could not measure the diversity between the generated and the original sentences, and propose the iBLEU variant by penalizing by the similarity with the original sentence. Therefore, we regard the iBLEU score as our major metric, which is also adpoted in Li et al. (2019).

In addition, we also conduct human evaluation in our experiments (detailed later).

[4] We used the code in https://github.com/timbmg/Sentence-VAE

Implementation Details

Our method involves unsupervised language modeling (forward and backward), realized by two-layer LSTM with 300 hidden units and trained specifically on each dataset with non-parallel sentences.

For hyperparameter tuning, we applied a grid search procedure on the validation set of the Quora dataset using the iBLEU metric. The power weights P, Q, and S in the objective were 8, 1, and 1, respectively, chosen from $\{0.5, 1, 2, \ldots, 8\}$.

The initial temperature T_{init} was chosen from $\{0.5, 1, 3, 5, 7, 9\} \times 10^{-2}$ and set to $T_{\text{init}} = 3 \times 10^{-2}$ by validation. The magnitude of T_{init} appears small here, but is in fact dependent on the scale of the objective function. The annealing rate C was set to $\frac{T_{\text{init}}}{\#\text{Iteration}} = 3 \times 10^{-4}$, where our number of iterations (#Iteration) was 100.

We should emphasize that all SA hyperparameters were validated only on the Quora dataset, and we did not perform any tuning on the other datasets (except the language model). This shows the robustness of our UPSA model and its hyperparameters.

Results

Table 1 presents the performance of all competing methods on the Quora and Wikianswers datasets. The unsupervised methods are only trained on the non-parallel sentences. The supervised models were trained on 100K paraphrase pairs for Quora and 500K pairs for Wikianswers. The domain-adapted supervised methods are trained on one dataset (Quora or Wikianswers) and tested on the other (Wikianswers or Quora).

We observe in Table 1 that, among unsupervised approaches, VAE achieves the worst performance on both datasets, indicating that paraphrasing by latent space sampling is worse than word editing. We further observe that UPSA yields significantly better results than CGMH: the iBLEU score of UPSA is higher than that of CGMH by 2–5 points. This shows that paraphrase generation is better modeled as an optimization process, instead of sampling from a distribution.

It is curious to see how our unsupervised paraphrase generator is compared with supervised ones, should large-scale parallel data be available. Admittedly, we see that supervised approaches generally outperform UPSA, as they can learn from massive parallel data. Our UPSA nevertheless achieves comparable results with the recent ResidualLSTM model (Prakash et al. 2016), reducing the gap between supervised and unsupervised paraphrasing.

In addition, our UPSA could be easily applied to new datasets and new domains, whereas the supervised setting does not generalize well. This is shown by a domain adaptation experiment, where a supervised model is trained on one domain but tested on the other. We notice in Table 1 that the performance of supervised models (e.g., Pointer-generator and Transformer+Copy) decreases drastically on out-of-domain sentences, even if both Quora and Wikianswers are question sentences. The performance is supposed to decrease further if the source and target domains are more

	Model	Quora				Wikianswers			
		iBLEU	BLEU	Rouge1	Rouge2	iBLEU	BLEU	Rouge1	Rouge2
Supervised	ResidualLSTM	12.67	17.57	59.22	32.40	22.94	27.36	48.52	18.71
	VAE-SVG-eq	15.17	20.04	59.98	33.30	26.35	32.98	50.93	19.11
	Pointer-generator	16.79	22.65	61.96	36.07	31.98	39.36	57.19	25.38
	Transformer	16.25	21.73	60.25	33.45	27.70	33.01	51.85	20.70
	Transformer+Copy	17.98	24.77	63.34	37.31	31.43	37.88	55.88	23.37
	DNPG	**18.01**	**25.03**	**63.73**	**37.75**	**34.15**	**41.64**	**57.32**	**25.88**
Supervised + Domain-adapted	Pointer-generator	5.04	6.96	41.89	12.77	21.87	27.94	53.99	20.85
	Transformer+Copy	6.17	8.15	44.89	14.79	23.25	29.22	53.33	21.02
	Shallow fusion	6.04	7.95	44.87	14.79	22.57	29.76	53.54	20.68
	MTL	4.90	6.37	37.64	11.83	18.34	23.65	48.19	17.53
	MTL+Copy	7.22	9.83	47.08	19.03	21.87	30.78	54.10	21.08
	DNPG	10.39	16.98	56.01	28.61	25.60	35.12	56.17	23.65
Unsupervised	VAE	8.16	13.96	44.55	22.64	17.92	24.13	31.87	12.08
	CGMH	9.94	15.73	48.73	26.12	20.05	26.45	43.31	16.53
	UPSA	12.02	18.18	56.51	30.69	24.84	32.39	54.12	21.45

Table 1: Performance on the Quora and Wikianswers datasets. The results of supervised learning and domain-adapted supervised methods are quoted from Li et al. (2019). We run experiments for all unsupervised methods and use the same evaluation script with Li et al. (2019) for a fair comparison. The results of CGMH in this table is slightly different from Miao et al. (2019), because Miao et al. (2019) use corpus-level BLEU, while Li et al. (2019) and we use sentence-level BLEU.

Model	MSCOCO				Twitter			
	iBLEU	BLEU	Rouge1	Rouge2	iBLEU	BLEU	Rouge1	Rouge2
VAE	7.48	11.09	31.78	8.66	2.92	3.46	15.13	3.40
CGMH	7.84	11.45	32.19	8.67	4.18	5.32	19.96	5.44
UPSA	**9.26**	**14.16**	**37.18**	**11.21**	**4.93**	**6.87**	**28.34**	**8.53**

Table 2: Performances on MSCOCO and Twitter.

Model	Relevance		Fluency	
	Mean Score	Agreement	Mean Score	Agreement
VAE	2.65	0.41	3.23	0.51
CGMH	3.08	0.36	3.51	0.49
UPSA	**3.78**	0.55	**3.66**	0.53

Table 3: Human evaluation on the Quora dataset.

different. UPSA outperforms all supervised domain-adapted paraphrase generators (except DNPG on the Wikianswers dataset), showing the generalizability of our model.

Table 2 shows model performance on MSCOCO and Twitter corpora. These datasets are less widely used for paraphrase generation than Quora and Wikianswers, and thus, we only compare unsupervised approaches by running existing code bases. Again, we see the same trend as Table 1: UPSA achieves the best performance, CGMH second, and the VAE worst. It is also noted that the Twitter corpus yields lower iBLEU scores for all models, which is largely due to the noise of Twitter utterances (Lan et al. 2017). However, the consistent results demonstrate that UPSA is robust and generalizable to different domains (without hyperparameter re-tuning).

Human Evaluation. We also conducted human evaluation on the generated paraphrases. Due to the limit of budget and resources, we sampled 300 sentences from the Quora test set and only compared the unsupervised methods (which is the main focus of our work). Selecting a subset of models and data samples is a common practice for human evaluation in previous work (Wang et al. 2019; Li et al. 2018).

We asked three human annotators to evaluate the generated paraphrases in terms of relevance and fluency; each aspect was scored from 1 to 5. We report in Table 3 the average human scores and the Cohen's kappa score (Cohen 1960).[5] It should be emphasized that our human evaluation was conducted in a blind fashion.

Table 3 shows that UPSA achieves the highest human satisfaction scores in terms of both relevance and fluency. The results are also consistent with the automatic metrics in Tables 1 and 2.

We further conducted two-sided Wilcoxon signed rank tests. The improvement of UPSA is statistically significant with $p < 0.01$ in both aspects, compared with both competing methods (UPSA vs. CGMH and UPSA vs. VAE).

Model Analysis

We analyze UPSA in more detail on the most widely-used Quora dataset, with a test subset of 2000 samples.

Ablation Study. We first evaluate the searching objective function (2) in Lines 1–4 of Table 5. The results show that each component of our objective (namely, keyword similarity, sentence similarity, and expression diversity) plays its role in paraphrase generation.

[5]According to McHugh (2012), a kappa score larger than 0.4 indicates moderate inter-annotator agreement.

Input	VAE	CGMH	UPSA
what would you do if given the power to become invisible ?	what would you do given the power to be invisible ? (4.33)	what do you do if given more power ? (3.33)	what would you do when you have a power to be invisible ? (4.67)
how can i become good in studies ?	how can i have a good android phone ? (2.33)	how can i become very rich in studies ? (4.00)	how should i do to get better grades in my studies ? (4.33)
what are the best colleges for mass communication in india ?	what are the best way of communication in india ? (2.67)	which are the top universities for mass marketing in india ? (3.67)	which is the top university for mass communication in india ? (4.33)
how does one avoid existential depression ?	how does one avoid belly fats ? (2.67)	how do i overcome my ocd ? (2.67)	how do you get over existential depression ? (4.33)
what are the pluses and minuses about life as a foreigner in singapore ?	what are the UNK and most interesting life as a foreigner in medieval greece ? (2.33)	what are the misconception about UNK with life as a foreigner in western ? (2.33)	what are the mistakes and pluses life as a foreigner in singapore ? (2.67)

Table 4: Example paraphrases generated by different methods on the Quora dataset. The averaged score evaluated by three annotators is shown at the end of each generated sentence.

Line #	UPSA Variant	iBLEU	BLEU	Rouge1	Rouge2
1	UPSA	**12.41**	18.48	57.06	31.39
2	w/o $f_{\text{sim,key}}$	10.28	15.34	50.85	26.42
3	w/o $f_{\text{sim,sen}}$	11.78	17.95	57.04	30.80
4	w/o f_{exp}	11.93	21.17	59.75	34.91
5	w/o copy	11.42	17.25	56.09	29.73
6	w/o annealing	10.56	16.52	56.02	29.25

Table 5: Ablation study.

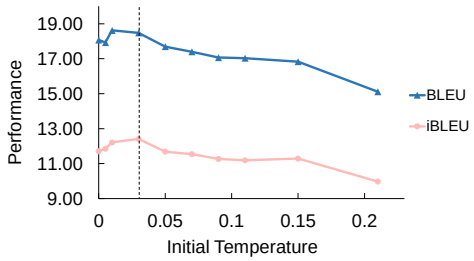

Figure 2: Analysis of the initial temperature T_{init}. The dashed line illustrates the selected hyperparameter in validation.

Line 5 of Table 5 shows the effect of our copy mechanism, which is used in word replacement and insertion. It yields roughly one iBLEU score improvement if we keep sampling those words in the original sentence.

Finally, we test the effect of the temperature decay in SA. Line 6 shows the performance if we fix the initial temperature during the whole searching process, which is similar to Metropolis–Hastings sampling.[6] The result shows the importance of the annealing schedule. It also verifies our intuition that sentence generation (in particular, paraphrasing in this paper) should be better modeled as a searching problem than a sampling problem.

Analysis of the Initial Temperature. We fixed the decreasing rate to $C = 1 \times 10^{-4}$ and chose the initial temperature T_{init} from $\{0, 0.5, 1, 3, 5, 7, 9, 11, 15, 21\} \times 10^{-2}$. In particular, $T_{\text{init}} = 0$ is equivalent to hill climbing (greedy search). The trend is plotted in Figure 2.

It is seen that a high temperature yields worse performance (with other hyperparameters fixed), because in this case UPSA accepts more worse sentences and is less likely to settle down.

On the other hand, a low temperature makes UPSA greedier, also resulting in worse performance. Especially, our simulated annealing largely outperforms greedy search, whose temperature is 0.

We further observe that BLEU and iBLEU peak at different values of the initial temperature. This is because a lower temperature indicates a greedier strategy with less editing, and if the input sentence is not changed much, we may indeed have a higher BLEU score. Our major metric iBLEU penalizes the similarity to the input and thus prefers a higher temperature. We chose $T_{\text{init}} = 0.03$ by validating on iBLEU.

Case Study. We showcase several generated paraphrases in Table 4. We see qualitatively that UPSA produces more reasonable paraphrases than VAE and CGMH in terms of both closeness in meaning and difference in expressions, even for the relatively long sentences. For example, "*if given the power to become invisible*" is paraphrased as "*when you have a power to be invisible*."

From the examples, we also observe that our current UPSA mainly synthesizes a paraphrase by editing words in the sentence, whereas the syntax of the original sentence is mostly preserved. This is partially due to the difficulty of exploring the entire (discrete) sentence space even by simulated annealing, and partially due to the insensitivity of the similarity objective given two very different sentences.

Conclusion

In this paper, we proposed a novel unsupervised approach UPSA that generates a paraphrase of a given sentence by simulated annealing. We propose a searching objective function, involving semantic preservation, expression diversity, and language fluency. We also propose a copy mechanism as our searching action. Experiments on four benchmark datasets show that our model outperforms previous state-of-the-art unsupervised methods to a large extent. We further outperform most domain-adaptive paraphrase generators, as well as a supervised model on the Wikianswers dataset.

In the future, we plan to apply the SA framework on syntactic parse trees in hopes of generating more syntactically different sentences (motivated by our case study).

[6]The Metropolis–Hastings sampler computes its acceptance rate in a different way from Eqn. (1).

References

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*, 936–945.

Barzilay, R., and Lee, L. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *ACL*, 16–23.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Jozefowicz, R.; and Bengio, S. 2016. Generating sentences from a continuous space. In *CoNLL*, 10–21.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Domhan, T., and Hieber, F. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *EMNLP*, 1500–1505.

Dong, Y.; Li, Z.; Rezagholizadeh, M.; and Cheung, J. C. K. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. *ACL* 3393–3402.

Edelkamp, S., and Schroedl, S. 2011. *Heuristic Search: Theory and Applications*. Elsevier.

Ellsworth, M., and Janin, A. 2007. Mutaphrase: Paraphrasing with framenet. In *Proc. ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 143–150.

Fader, A.; Zettlemoyer, L.; and Etzioni, O. 2013. Paraphrase-driven learning for open question answering. In *ACL*, 1608–1618.

Granville, V.; Krivanek, M.; and Rasson, J. 1994. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(6):652–656.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, 1631–1640.

Gulcehre, C.; Firat, O.; Xu, K.; Cho, K.; Barrault, L.; Lin, H.-C.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Gupta, A.; Agarwal, A.; Singh, P.; and Rai, P. 2018. A deep generative framework for paraphrase generation. In *AAAI*, 5149–5156.

Guu, K.; Hashimoto, T. B.; Oren, Y.; and Liang, P. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics* 6:437–450.

Hwang, C.-R. 1988. Simulated annealing: theory and applications. *Acta Applicandae Mathematicae* 12(1):108–111.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by simulated annealing. *Science* 220(4598):671–680.

Knight, K., and Marcu, D. 2000. Statistics-based summarization step one: Sentence compression. In *AAAI*, 703–710.

Lan, W.; Qiu, S.; He, H.; and Xu, W. 2017. A continuously growing dataset of sentential paraphrases. In *EMNLP*, 1224–1234.

Li, Z.; Jiang, X.; Shang, L.; and Li, H. 2018. Paraphrase generation with deep reinforcement learning. In *EMNLP*, 3865–3878.

Li, Z.; Jiang, X.; Shang, L.; and Liu, Q. 2019. Decomposable neural paraphrase generation. In *ACL*, 3403–3414.

Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollar, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common objects in context. In *ECCV*, 740–755.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. Workshop on Text Summarization Branches Out*, 74–81.

Lowerre, B. T., and Reddy, D. R. 1980. The harpy speech recognition system. In *Trends in Speech Recognition*.

McHugh, M. L. 2012. Interrater reliability: The kappa statistic. *Biochemia Medica* 22(3):276–282.

Mckeown, K. R. 1983. Paraphrasing questions using given and new information. *Computational Linguistics* 9(1):1–10.

Metropolis, N.; Rosenbluth, A. W.; Rosenbluth, M. N.; Teller, A. H.; and Teller, E. 1953. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21(6):1087–1092.

Miao, N.; Zhou, H.; Mou, L.; Yan, R.; and Li, L. 2019. Constrained sentence generation by Metropolis–Hastings sampling. In *AAAI*, 6834–6842.

Narayan, S.; Reddy, S.; and Cohen, S. B. 2016. Paraphrase generation from latent-variable for semantic parsing. In *INLG*, 153–162.

Neidert, J.; Schuster, S.; Green, S.; Heafield, K.; and Manning, C. 2014. Stanford University's submissions to the WMT 2014 translation task. In *Proc. 9th Workshop on Statistical Machine Translation*, 150–156.

Pagliardini, M.; Gupta, P.; and Jaggi, M. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL*, 528–540.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 311–318.

Pennington, J.; Socher, R.; and Manning, C. 2014. GloVe: global vectors for word representation. In *EMNLP*, 1532–1543.

Prakash, A.; Hasan, S. A.; Lee, K.; Datla, V.; Qadir, A.; Liu, J.; and Farri, O. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *COLING*, 2923–2934.

Rose, S.; Engel, D.; Cramer, N.; and Cowley, W. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory* 1:1–20.

Roy, A., and Grangier, D. 2019. Unsupervised paraphrasing without translation. In *ACL*, 6033–6039.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Shah, P.; Hakkani-Tür, D.; Tür, G.; Rastogi, A.; Bapna, A.; Nayak, N.; and Heck, L. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Sun, H., and Zhou, M. 2012. Joint learning of a dual SMT system for paraphrase generation. In *ACL*, 38–42.

Van den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. In *NIPS*, 6306–6315.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Wang, S.; Gupta, R.; Chang, N.; and Baldridge, J. 2019. A task in a suit and a tie: Paraphrase generation with semantic augmentation. In *AAAI*, 7176–7183.

Wieting, J., and Gimpel, K. 2017. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*, 451–462.

Zhao, S.; Lan, X.; Liu, T.; and Li, S. 2009. Application-driven statistical paraphrase generation. In *ACL*, 834–842.