Towards Neural Controllable Text Generation

LI, Jingjing

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
August 2023

Thesis Assessment Committee

Professor YOUNG Fung Yu (Chair)

Professor KING Kuo Chin Irwin (Thesis Supervisor)

Professor LYU Rung Tsong Michael (Thesis Co-supervisor)

Professor LAM Wai (Committee Member)

Professor LIN Shou-De (External Examiner)

Abstract of thesis entitled:

Towards Neural Controllable Text Generation

Submitted by LI, Jingjing

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in August 2023

Text generation is the task of generating textual contents which are indistinguishable from natural language. With the advance of deep learning techniques, neural network models achieve giant success and become the dominating solution to the task of text generation. Despite the remarkable success in writing fluent and creative content, existing solutions pay little attention to the controllability over the generation systems. While in practice, there is a substantial need to steer a powerful neural generation system to deliver output text that satisfies diverse attributes, such as style, sentiment and topic.

In this thesis, we explore a novel stream of text generation problem: controllable text generation. This task aims to generate textual contents that meet the target attribute requirements predefined by human beings, allowing for greater controllability over the output text. Notably, it is observed that the training procedure of neural controllable text generation models is quite data-intensive and thus often suffer from the data-scarcity issue. On account of this observation, in this thesis, we further dive into the problem of neural controllable text generation (NCTG) under bipartite settings: training a NCTG model (i) with full supervision and (ii) with no supervision from large parallel corpora.

First, we study the semantic fidelity control in neural text gen-

eration. Question generation (QG) is a fundamental task in the field of text generation. The target is to generate a question from a reference sentence and a specified answer within the sentence. A sound QG system has a high demand of semantic fidelity over its output. Existing sequence-to-sequence neural models achieve this goal by proximity-based answer position encoding under the intuition that neighboring words of answers are of high possibility to be answer-relevant. However, such intuition may not apply to all cases, especially for sentences with complex answer-relevant relations. Consequently, the performance of these models drops sharply when the relative distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question increases. To address this issue, we propose a method to jointly model the unstructured sentence and the structured answer-relevant relation (extracted from the sentence in advance) for QG. Specifically, the structured answer-relevant relation acts as the to the point context and it thus naturally helps keep the generated question to the point, while the unstructured sentence provides the full information. Extensive experiments show that to the point context helps our QG model achieve significant improvements on several automatic evaluation metrics. Furthermore, our model is capable of generating diverse questions for a sentence which conveys multiple relations of its answer fragment.

Secondly, we investigate NCTG in an unsupervised setting. We propose TGLS, a novel framework for 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 and satisfaction score of sentences. Then, a conditional generative model learns from the search results, and meanwhile smooths 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 text generation tasks,

unsupervised paraphrasing and text formalization. Our model significantly outperforms unsupervised baseline methods in both tasks. Especially, it achieves comparable performance to strong supervised methods for paraphrase generation.

Finally, we dive into the efficiency issue of current solutions to unsupervised NCTG. In the field of text generation, there is a family of revision tasks where the source and target sequences share moderate resemblance in surface form but differentiate in attributes, such as text formality and simplicity. Current state-of-the-art methods formulate these tasks as sequence-to-sequence learning problems, which rely on large-scale parallel training corpus. In this project, we present an iterative in-place editing approach for text revision, which requires no parallel data. We simply fine-tune a pre-trained Transformer with masked language modeling and attribute classification. During inference, the editing at each iteration is realized by two-step span replacement. At the first step, the distributed representation of the text optimizes on the fly towards an attribute function. At the second step, a text span is masked and another new one is proposed conditioned on the optimized representation. The empirical experiments on two typical and important text revision tasks, text formalization and text simplification, show the effectiveness of our approach. It achieves competitive and even better performance than state-of-the-art supervised methods on text simplification, and gains better performance than strong unsupervised methods on text formalization.

論文題目：基於神經網絡的可控性文本生成

作　　者：李菁菁

學　　校：香港中文大學

學　　繫：計算機科學與工程學繫

修讀學位：哲學博士

摘　　要：

文本生成是一項生成與自然語言貼近的文字的任務。隨著深度學習技術的進步，神經網絡模型取得了巨大的成功，在文本生成任務中成為了主導解決方案。儘管現有的解決方案在撰寫流暢、有創意的內容上取得了顯著的成功，但在生成繫統的可控性上缺乏深入研究。實際中，大量的應用場景需要引導文本生成繫統產生滿足不同屬性要求的輸出文本，例如風格、情感和主題。文本生成繫統的可控性是一個相當重要的功能。

在本論文中，我們探討了一種新的文本生成問題：可控文本生成。這個任務的目標是生成滿足人類預先定義的目標屬性要求的文本內容，從而實現對輸出文本的控製。值得註意的是，目前基於神經網絡方法的可控文本生成模型的訓練程序非常依賴大規模的數據，因此常常受到數據稀缺問題的影響。鑒於這一觀察結果，在本論文中，我們從多個角度深入研究了基於神經網絡的可控文本生成（NCTG）問題：（1）使用大規模平行預料，以有監督的方式訓練可控的生成模型，以及（2）使用無標註的非平行語料，以無監督方式搭建可控文本生成繫統。

首先，我們研究神經文本生成中的語義準確度控製問題。問題生成（QG）是文本生成領域的一項重要任務。其目標是根據參考句子和指定答案生成相應問題。一個好的問題生成繫統對其輸出的語義準確度要求很高。現有的序列到序列神經模

型通過基於鄰近答案位置編碼的方式來實現此目標，因為答案附近的詞語很可能是問題高度相關的。然而，對於具有複雜答案相關關係的句子，此類假設可能不適用。因此，當答案詞與其他非停用詞的上下文之間的相對距離增加時，這些模型的性能會急劇下降。為了解決這個問題，我們提出了一種方法，將非結構化的句子和事先從句子中提取的結構化答案詞相關上下文聯合建模，用於輔助問題生成繫統生成和答案詞上下文高度相關的問題。具體來講，將結構化的答案詞相關的語境信息作為要點上下文輸入給生成繫統，自然可以幫助生成和要點信息語義高度關聯的問題；同時非結構化的句子可以提供完整的上下文信息。大量實驗錶明，要點上下文有助於我們的模型生成和輸入信息高度關聯且語義準確的問題，在幾個自動評估指標上的錶現都有顯著提高。此外，我們的模型能夠為包含多個答案詞相關的語境信息的句子生成相應的多樣的問題。

其次，我們在無監督的場景下研究了可控文本生成問題。我們提出了新的框架TGLS，一個在句子空間中進行蒐索併學習的無監督文本生成方案。我們首先將一個强大的蒐索算法，模擬退火算法，應用於一個啓發式定義的目標函數上。該目標函數估計了蒐索中的得到的候選句的質量併提供滿意度打分。然後，一個條件生成模型從蒐索的結果中進行學習，同時平滑蒐索中的得到的噪聲。最終繫統的錶現可以通過交替進行句子空間蒐索和生成模型學習來進一步提高。我們在兩個現實世界的可控的文本生成生成任務上展示了TGLS的有效性，包括無監督的復述生成和文本正式化。我們的模型在這兩個任務中都明顯優於無監督基線方法。特別地，它在文本復述方麵實現了與强監督方法相當的性能。

最後，我們深入探討了當前解決無監督場景下的可控文本生成問題的效率問題。在文本生成領域中，有一類修訂任務，其中源序列和目標序列在錶麵形式上具有相對較高的相似度，但在屬性上卻有所不同，例如文本正式化和文本簡化。當前最先進的方法是將這些任務定義為序列到序列學習問題，依賴於大規模的平行訓練語料庫。在本項目中，我們提出了一種文本修訂的疊代式原地編輯方法，它不需要依賴於標註好的平行數

據。我們隻需要使用遮蔽語言建模和屬性分類兩個訓練目標對預訓練過的Transformer模型進行微調。在推理過程中，每一次的編輯通過兩步實現。第一步，文本的分佈式錶示會即時優化，隱藏層參數嚮目標屬性函數靠攏。第二步，基於優化後的隱藏層參數，待修改的一段文本段會被遮蔽，併替換成另提出的一段符合目標屬性的新文本。通過文本正式化和文本簡化這兩個典型的文本修訂任務的實驗，我們提出的方法在屬性控製上取得了良好的效果，甚至優於現有的强監督方法和無監督方法。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The thesis expounds our research on neural controllable text generation. We first endeavor to provide a broad overview of this problem in Section 1.1. Our contributions to this specific domain are elucidated in Section 1.2. To conclude, the ensuing Section 1.3 presents a concrete outline of this thesis.

1.1 Motivation

Natural language processing (NLP) is an interdisciplinary field of study within the realm of computer science and linguistics that focuses on enabling machines to understand, interpret, and generate natural language. It involves developing algorithms and methodologies that enable computers to analyze, process, and

manipulate large volumes of unstructured human language data, such as text and speech.

One of the most exciting subfield of NLP is text generation. Text generation, or natural language generation (NLG), refers to the technique of automatically generating fluent contents which is coherent and contextually relevant to the input instructions. It is the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information [4]. In recent years, deep neural network techniques have shown great promise in generating high-quality text, and has been the go-to solution to text generation [5, 6, 7, 8].

Despite the significant progress made in this field, neural text generation also suffers from several drawbacks. One of the major issues with neural text generation is the lack of control over the generated text. For instance, a text generator usually inherits everything from training data, which might contain harmful or biased contents [9, 10, 11]. Without appropriate controlling methods, this vanilla text generator might lead to the propagation of

toxic content. When scaled up in size, Large Language Models (LLMs) [12, 13] demonstrate incredible abilities in text completion and result in the prevalence of text generation applications. Ensuring the secure deployment of LLMs becomes more urgent. Furthermore, neural text generation models tend to replicate the patterns in the training data, which can result in a lack of versatility in the generated content. From the perspective of linguistics, conveying appropriate messages that are adequate to the social context highly depends on controlling the style of a text [14, 15, 16]. This gives rise to an emerging research area: controllable text generation (CTG).

Controllable text generation (CTG) is to create a language generation system that can be directed towards generating texts with desired properties, while still maintaining the overall coherence and naturalness. Compared with text generation, CTG enables users to specify certain constraints or preferences on the generated text. Typically, the text generator component in CTG usually conditions a user-specified control code. The predetermined textual attributes or characteristics can range from style [17, 18], sentiment [19, 20], formality [21, 22, 23], syntax [24, 25] to persona [26], topic [27],

and so on.

As discussed, making text generation more controllable is an essential problem in NLP. CTG achieves controllability by allowing for greater control over the content and style of the generated text. With this technique, we are able to regulate the text generation procedure to mitigate the generation of potentially harmful content. Moreover, there is a substantial amount of attributes to choose from. The flexibility of choices enables users to tailor the generation system according to their own needs like news digesting for users with different knowledge backgrounds. Furthermore, despite satisfying the user-specific attribute requirements, controllable text generation models are also suitable for many task-specific applications, such as writing assistance tools, personalized conversation agents, and stylized creative writing.

Leveraging the expressive capability of neural network models [28, 29, 30, 31], controllable text generation has exhibited nonnegligible potential in various downstream applications. In this thesis, we focus on the paradigm of neural controllable text generation (NCTG). Aside from its wide applications, there are some challenges hindering the rapid advance of NCTG. One vi-

tal problem of neural approaches is that the output text can be unfaithful to the source input. Sometimes the model inclines to generate incoherent or hallucinated or biased contents [9, 32, 33]. Such performance poses a risk to the reliability and factuality of the generated content. Therefore, controlling the semantic fidelity of a text generation system is of great importance.

Another drawback of existing neural approaches is that a successful training practice highly depends on a large-scale high-quality labeled dataset. Training a single-attribute NCTG model demands such an amount of data, finding appropriate training data for a multi-attribute NCTG model becomes even harder. And it is infeasible to address the problems in a low-resource setting. It inspires our research in two dimensions: (1) how to generate high-quality datasets? (2) how to directly achieve the attribute transfer without using parallel corpus?

Additionally, developing an NCTG model also demands substantial computing resources. With the increasing size of language models, pre-training a language model conditioned on multiple attributes is quite computationally expensive. For example, to fine-tune a GPT-3 summarization model with Reinforcement Learning

(RL), it takes thousands of labeler hours for learning a reliable reward function and 320 GPU-days to train the policy and value nets [34].

In this thesis, we are dedicated to addressing the above challenges of neural controllable text generation. Our research of controllable text generation consists of bipartite machine learning settings: fully supervised learning and unsupervised learning. In the first part, we start with a core issue in neural text generation and the classical setting in machine learning: to control the semantic fidelity of neural text generation in a supervised manner. In the second part, our research involved diverse attributes in a more challenging setting, where we study the control lexical diversity, formality, and simplicity in neural text generation in an unsupervised manner.

1.2 Thesis Contributions

As presented above, the research focus of this thesis is to find applicable solutions to neural controllable text generation under bipartite machine learning settings. In the initial phase, our study is to regulate the semantic fidelity of neural text generation. As

part of this endeavor, we adopt the task of question generation as the experimental benchmark for our study. Subsequently, in the second phase, our research investigates various attributes in a more complex setting. In this context, we explore different sub-tasks in neural text generation, including lexical diversity in the task of paraphrase generation, formality control in text formalization, and simplicity control in text simplification, employing an unsupervised setting. Our study covers a wide range of controllable text attributes and provides solutions to CTG in different settings.

- To study the control of semantic fidelity, we propose a novel framework to capture the to-the-point context in the source text and use it to guide the generation of target text in the task of question generation (QG). We analyze the existing models and conclude that their proximity-based answer position encoding is not globally applicable, especially for sentences with free structures. To ensure that the generated question is semantically relevant to the input answer and passage, we extract the structured answer-relevant relation and design an advanced multi-encoder to jointly model the unstructured sentence and the structured answer-relevant relation for question

generation. Extensive experiments indicate that our method can improve the semantic correctness of generated questions by steering the generation system to attend to the extracted context restriction.

- To explore the text attribute transfer in the unsupervised setting, we introduce TGLS, a principled framework for unsupervised text generation. In general, TGLS can be applied to different tasks if the output resembles the input and can be roughly estimated by a heuristically defined scoring function. Additionally, we successfully incorporate large-scale pretrained language models into our TGLS framework. We conduct experiments on two different attributes: lexical diversity in paraphrasing and formality in text formalization. In both experiments, TGLS significantly outperforms unsupervised baseline methods. Moreover, TGLS achieves comparable performance to recent supervised models in the paraphrasing task. 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.

- To investigate the manipulation of text attributes via un-

supervised revision, we present an efficient mask-and-infill method with on-the-fly optimized representation for text revision. This framework is readily adaptable to other textual attributes. To enable on-the-fly representation optimization, we design simple fine-tuning methods that balance efficiency and efficacy. The fine-tuning can be finished within 8 GPU-hours at most in our experiments. Specifically, we address the revision of two important attributes: text simplicity and formality. Our proposed OREO has strong performance on text formalization dataset GYAFC-fr, surpassing unsupervised baseline methods, one of which also utilizes RoBERTa; and achieves competitive performance with state-of-the-art supervised methods on text simplification dataset NEWSELA-TURK.

1.3 Thesis Outline

In this thesis, we will present our investigation of neural controllable text generation. The remaining part of this thesis is structured as follows:

Chapter 2 presents a background review of NCTG. We start

with the introduction of preliminary knowledge of deep learning techniques in NLP in Section 2.1. Then we provide the literature review of some representative tasks under CTG in Section 2.2, including question generation, paraphrase generation, text style transfer, and text simplification. At last, we elaborate on the evaluation paradigm of CTG tasks in Section 2.3.

Chapter 3 demonstrates our exploration of controlling the semantic fidelity of neural text generation in the task of question generation. We first provide an introduction to question generation in Section 3.1 and the formalization of this problem in Section 3.2. The technical details of our approach are covered in Section 3.3. We perform exhaustive experiments in Section 3.4 and further analysis is provided in Section 3.5. Finally, this work is concluded in Section 3.6.

Chapter 4 delves into the task of CTG in the unsupervised setting. We initiate with an overview of unsupervised text generation and CTG in Section 4.1. Section 4.2 encompasses the technical aspects of our proposed approach, followed by comprehensive experiments in Section 4.3 on paraphrase generation and text formalization. The empirical results and additional analysis

can be found in Section 4.4. Section 4.5 summarizes this work.

Chapter 5 highlights our examination of unsupervised NCTG in the task of text revision. We begin with an introductory discussion on text revision in Section 5.1 and then proceed with the problem's formalization in Section 5.2. The technical components of our approach are discussed in Section 5.3, and Section 5.4 details the thorough experiments in this work. An in-depth analysis is provided in Section 5.5. This work is wrapped up in Section 5.6.

Chapter 6 concludes with a summary of the thesis in Section 6.1, followed by a discussion of potential research directions, including the exploration of structured control codes and the method to deal with continuously increasing control codes in Section 6.2.

□ **End of chapter.**

Chapter 2

Background Review

In this chapter, we present the foundational knowledge and relevant studies of neural controllable text generation. We first provide the background knowledge of deep learning techniques in Section 2.1, with a coverage of the network structure of popular neural networks for modeling language, and the pretraining techniques. Section 2.2 introduces some representative tasks in the field of NCTG and their related work. Each of the tasks focuses on a distinct control factor of text, including question generation, paraphrase generation, text style transfer, and text simplification. Section 2.3 presents a summary of the evaluation metrics utilized in neural controllable text generation.

Our contribution in this field is positioned within a taxonomy

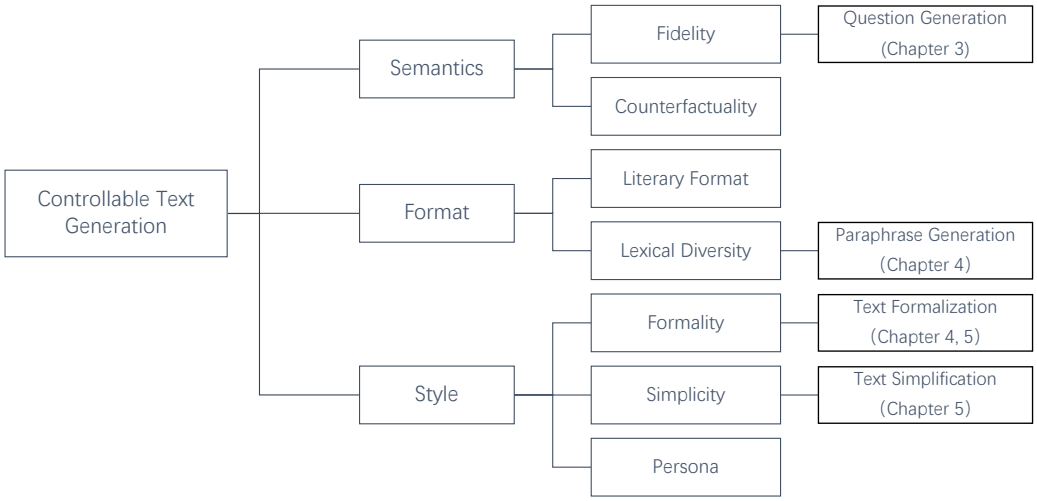

Figure 2.1: The taxonomy of neural controllable text generation. Per control codes, we broadly categorize the existing research on NCTG into three types: semantics, format and style. The bold leaf nodes locate our contributions.

of NCTG, as shown in Figure 2.1. This taxonomy provides a comprehensive overview of the research directions in NCTG and highlights the unique characteristics of our work in comparison to previous research.

2.1 Deep Learning Fundamentals of NLP

With the remarkable efficacy of a wide range of intricate tasks, deep learning techniques have been the go-to solution in various fields. In this section, we will introduce the background knowledge of deep learning techniques applied in NLP applications. We first

introduce the approach to language modeling and the architecture of Recurrent Neural Network (RNN) in the first part. Then we move to the dominant solution to language generation, i.e., the sequence-to-sequence paradigm. In the end, we will dive into the basics of an emerging neural architecture, Transformer, and the pretraining methods which elevate the development of NLP to a more advanced level.

2.1.1 Language Modeling and Recurrent Neural Networks

Text, as one of the realization surfaces that human beings convey and exchange information, consists of sequentially-structured data. In order to model the dynamics of sequentially formatted data, a language model is often used to estimate the joint probability of occurrence of a number of words in a particular sequence. Given a text sequence S with n words, $S = \{x_1, x_2, ..., x_n\}$, we denote the joint probability of this sequence as $P(x_1, ..., x_n)$. Language modeling plays a crucial role in NLP applications. One important task is to compare the quality of texts generated by different systems. We are able to resolve this problem by using the likelihood

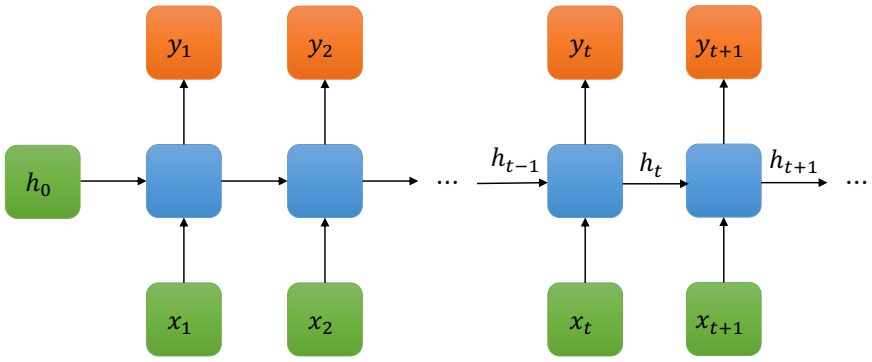

Figure 2.2: The architecture of a RNN model.

predicted by the language models.

Despite the capability to compare likelihood, a language model can be used to sample textual sequences. In a textual sequence, the selection of i_{th} word often depends on the context of the first $(i-1)_{th}$ words. Therefore, the joint probability $P(x_1, ..., x_n)$ can be factorized into the product of all words, each of which is conditioned on all previous contextual words:

$$P(x_1, ..., x_n) = \prod_{t=1}^{n} P(x_t | x_1, ..., x_{t-1}). \tag{2.1}$$

With this, a language model can generate natural text by sequentially drawing word x_t from $P(x_t|x_1, ..., x_{t-1})$.

Recurrent Neural Network is proposed to efficiently model the conditional probability $P(x_t|x_1, ..., x_{t-1})$. The input text is rep-

resented as a sequence of fix-length vectors $S = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$, $\mathbf{x}_t \in \mathcal{R}^d$. At each step, RNN takes in the output of the previous layer $\mathbf{h}_{t-1}$ and the input distributed word representation $\mathbf{x}_t$. $\mathbf{h}_{t-1}$ represents the hidden states of timestep $t-1$. To calculate the hidden state of current timestep, it first performs linear transformation to both input and a non-linear transformation after that. Specifically, both input elements are multiplied by two weight matrices $\mathbf{W}_{hh}$, $\mathbf{W}_{hx}$ respectively, followed by a sigmoid function (Equation 2.2). Then the output feature $\mathbf{h}_t$ is multiplied by a weight matrix $\mathbf{W}_o$ and passes through a softmax to obtain the vocabulary distribution for the prediction of word x_t (Equation 2.3).

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_t), \tag{2.2}$$

$$P(x_t|x_1, ..., x_{t-1}) = softmax(\mathbf{W}_o\mathbf{h}_t). \tag{2.3}$$

An architecture of a RNN architecture is illustrated in 2.2. On the basis of the original RNN model, there is a substantial amount of follow-up innovations of more advanced RNN architectures. To address the gradient vanishing issue in earlier recurrent networks, the memory mechanism was introduced in Long Short-Term Mem-

ory (LSTM) [28] and its light-weight version, Gated Recurrent Units (GRU) [30]. To improve long-term memory, Bidirectional Recurrent Neural Networks (BiRNN) [29] was proposed to encode sequential information from both the past and future. These works have exhibited giant success in multiple applications.

2.1.2 Sequence-to-Sequence Paradigm

RNN is an effective tool for estimating the likelihood of text or generating text. While in the field of text generation, many tasks incline to learn the mapping function between two unaligned, sequentially structured text. For example, in paraphrase generation, given an input sequence, the target is to generate a sequence that has diverse syntax and lexicons with the input one. In general, these problems are formalized as sequence-to-sequence (seq2seq) problems [35].

The solution to the seq2seq problem is the seq2seq paradigm. The seq2seq paradigm follows an encoder-decoder structure, where the encoder encodes the information of a sequentially structured text with variable length and the decoder receives the encoded input information to predict the expected output as a conditional

language model. The encoder and decoder can be implemented as RNNs or any other neural network architecture, depending on the task formalization.

In text generation, the encoder RNN usually takes in a varying-length input sequence and transforms it into a context vector. Given input sentence $S = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$, RNN encoder transforms each word $\mathbf{x_t}$ to hidden state $\mathbf{h}_t^{enc}$:

$$\mathbf{h}_t^{enc} = f_{enc}(\mathbf{h}_{t-1}^{enc}, \mathbf{x_t}). \tag{2.4}$$

Then we can obtain the context vector $\mathbf{c}$ through customized function g:

$$\mathbf{c} = g(\mathbf{h}_1^{enc}, ..., \mathbf{h}_n^{enc}). \tag{2.5}$$

The decoder is a separate RNN model taking charge of generating the target sequence $Y = \{y_1, y_2, ..., y_m\}$ word by word. The prediction of each word y_t is conditioned on the input sequence and the preceding decoded words. Different from encoder RNN, decoder RNN computes the hidden states at time-step t by taking in three components, the hidden states in last time-step $\mathbf{h}_{t-1}^{dec}$, word

prediction $\hat{\mathbf{y}}_{t-1}$ and the context vector $\mathbf{c}_t$ as follows:

$$\mathbf{h}_t^{dec} = f_{dec}(\mathbf{h}_{t-1}^{dec}, \hat{\mathbf{y}}_{t-1}, \mathbf{c}_t). \tag{2.6}$$

The hidden states at time-step t are passed through an output layer and a softmax function to compute the vocabulary distribution of target word y_t. This process is similar to a common language model.

2.1.3 Transformer Architecture and Pretraining

Despite the splendid breakthroughs of RNN in the past, recent progress in natural language processing has been dominantly boosted by the Transformer [36] architecture. The first proposal of Transformer targets at sequence-to-sequence learning and outperforms the best previously reported approaches by a large margin in machine translation. Now it has been widely applied to various deep learning applications other than text transduction, such as computer vision [37] and speech [38].

The basic element in Transformer architecture is the self-attention mechanism. The attention mechanism was originally designed for encoder-decoder RNN architecture in seq2seq problems[39, 40].

As for self-attention, the vanilla Transformer takes the scales dot-product attention [36]. Consider a token x_i in the sentence $S = \{x_1, x_2, ..., x_n\}$, its query, key and value are denoted as $\mathbf{q}_i \in \mathcal{R}^d$, $\mathbf{k}_i \in \mathcal{R}^d$ and $\mathbf{v}_i \in \mathcal{R}^d$. The stacked query, key and value matrix in the sentence are denoted as $\mathbf{Q} \in \mathcal{R}^{n\times d}$, $\mathbf{K} \in \mathcal{R}^{n\times d}$ and $\mathbf{V} \in \mathcal{R}^{n\times d}$, respectively. Then the self-attention is computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}. \tag{2.7}$$

The vanilla Transformer consists of a transformer encoder and a transformer decoder, each of which encompasses a stack of six identical layers. Each layer is comprised of a multi-head attention sub-layer and a point-wise feed-forward network. Inspired by [41], a residual connection is employed at the top of both sub-layers, followed by layer normalization. The decoder has an additional multi-head attention sub-layer, which is to calculate the encoder-decoder attention.

Despite the remarkable performance in many NLP tasks, the self-attention mechanism endows Transformer with the capability of parallelized computation, with a large reduce in the computation complexity. In the practice of pretraining, Transformer also demon-

strates its scalability [42]. Therefore, many large-scale pretrained models adopt the Transformer block as their base architecture.

Pretraining techniques and pretrained models have revolutionized NLP research by enabling transfer learning and significantly improving the performance of various downstream tasks. The pretraining techniques involve training large-scale language models on vast amounts of unlabeled text data, allowing them to learn rich representations of language that capture syntactic, semantic, and contextual information.

In the era of pretraining, the large-scale Transformer-based models are firstly optimized with pretraining objectives. Then the pretrained models are fine-tuned with task-specific corpora when applied to downstream tasks. Due to the reliance on a large volume of training data, most pretraining tasks follow the paradigm of self-supervised learning. The commonly used pretraining tasks include language modeling [43, 44, 45], masked language modeling [46, 47, 48], permutated language modeling [49] and denoising autoencoder [50].

According to the model structure, existing pretrained models can be categorized into three types, (1) **encoder-only**: Bidirec-

tional Encoder Representations from Transformers (BERT) [46], Generalized Autoregressive Pretrained Transformer (XLNET) [49], Robustly Optimized BERT (RoBERTa) [47]; (2) **encoder-decoder**: Denoising Sequence-to-Sequence Pre-trained Transformer (BART) [50], Text-to-Text Transfer Transformer (T5) [48]; and (3) **decoder-only**: Generative Pretrained Transformer (GPT) [44], GPT-2 [45], GPT-3 [51].

The research on pretrained models has paved the way for transfer learning and has become valuable resource for researchers and practitioners in the field. They provide powerful and general-purpose language representations that can be fine-tuned for specific tasks, leading to improved performance and efficiency.

2.2 Representative Tasks Involving CTG

In this section, we introduce some representative tasks with distinct control aspects in the field of CTG. As exhibited in Figure 2.1, based on different control factors, research in controllable text generation can be broadly classified into three categories: semantics, format, and style.

- Semantics: The control factors in this area are related to the

semantic aspects of the input, such as fidelity and counterfactuality. An example application that demands semantic relevance between input reading materials and generated text is question generation.

- Format: This category involves controlling the surface realization of generated text, such as adhering to the structures of specific literary (literacy format) or ensuring the output text has different wordings compared to the input text (lexical diversity). Paraphrase generation is a representative task of the latter one.

- Style: This category focuses on controlling the expressive style of the text, such as incorporating the personality of a character (persona), maintaining a formal tone (formality) and simplified usage of the overall language and vocabulary used in the text (simplicity). The corresponding research tasks for the last two are text formalization and text simplification, respectively.

In this section, we will primarily introduce the task definitions and provide an overview of relevant research regarding the tasks of

question generation, paraphrase generation, and text formalization and text simplification, which will be discussed in Section 2.2.1, Section 2.2.2, Section 2.2.3 and Section 2.2.4, respectively.

2.2.1 Question Generation

Question generation (QG) is an essential task in neural text generation. Given input text and answer words, neural question generation aims to generate meaningful questions [2, 3]. Since the generated questions are expected to be contextually relevant to the input, controlling the generation system to focus on pertinent contexts of the input text is of great importance. The question generation task can be formally defined as follows:

Definition 1. *Let $\mathcal{X}$ be the input text, represented as a sequence of words, $\mathcal{A}$ be the answer span within the input, and $\mathcal{Y}$ be the generated question. The goal of question generation is to learn a mapping function f: $\mathcal{X} \rightarrow \mathcal{Y}$, where f predicts coherent and contextually relevant questions $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ given an input $\mathcal{X} = \{x_1, x_2, ..., x_m\}$ and the answer span $\mathcal{A} = \{a_1, a_2, ..., a_i\}$. The mapping function f is parameterized by a set of model parameters θ. The generation process is formulated as maximizing the conditional*

probability $P(\mathcal{Y}|\mathcal{X}, \mathcal{A}; \theta)$*, which refers to obtaining* $\mathcal{Y}$ *conditioning on input* $\mathcal{X}$ *and model parameters.*

Initially motivated for educational purposes, the task of question generation is tackled by designing many complex rules for specific question types [52, 53]. Heilman and Smith [54] improve rule-based question generation by introducing a statistical ranking model. First, they remove extraneous information in the sentence to transform it into a simpler one, which can be transformed easily into a succinct question with predefined sets of general rules. Then they adopt an overgenerate-and-rank approach to select the best candidate considering several features.

With the rise of dominant neural sequence-to-sequence learning models [35], Du et al. [2] frame question generation as a sequence-to-sequence learning problem. Compared with rule-based approaches, neural models [55] can generate more fluent and grammatical questions. However, question generation is a one-to-many sequence generation problem, i.e., several aspects can be asked given a sentence, which confuses the model during train and prevents concrete automatic evaluation. To tackle this issue, Zhou et al. [3] propose the answer-aware question generation setting which

assumes the answer is already known and acts as a contiguous span inside the input sentence. They adopt a BIO tagging scheme to incorporate the answer position information as learned embedding features in seq2seq learning. In the BIO tagging scheme, B denotes the starting word of an answer span and I indicates the continuing answer words. Through this scheme, Zhou et al. [3] encodes the BIO tags in the position embeddings to inform the positional information of the answer span to the RNN encoder. Song et al. [8] explicitly model the information between answer and sentence with a multi-perspective matching model. Kim et al. [56] also focus on the answer information and proposed an answer-separated seq2seq model by masking the answer with special tokens. Besides the BIO tagging scheme, the binary indicator is also a well-adopted approach to encode the answer position information. Yang et al. [57] and Yuan et al. [55] incorporate the binary features to word embeddings of tokens to differentiate the context words and answer words in one paragraph or a document, respectively. All answer-aware neural models treat question generation as a one-to-one mapping problem, but existing models perform poorly for sentences with a complex structure.

Heilman and Smith [54] and Cao et al. [58] address the problem by removing the extraneous information. Heilman and Smith [54] directly use the simplified sentence for generation, while Cao et al. [58] consider aggregating two sources of information via gated attention in summarization. Factoid question generation from structured text is initially investigated by Serban et al. [59], but our focus here is leveraging structured inputs to help question generation over unstructured sentences. Our proposed model in Chapter 3 can take advantage of unstructured sentences and structured answer-relevant relations to maintain informativeness and faithfulness of generated questions.

2.2.2 Paraphrase Generation

Paraphrase generation refers to rephrasing the input text with different surface forms [60]. In essence, paraphrasing involves the manipulation of lexicon during the generation process to create text that conveys the same meaning while using different words and structures. Table 2.1 provides some examples of sentence-level paraphrasing. The formal definition for paraphrase generation is as below:

Table 2.1: Examples of paraphrases from benchmark dataset ParaNMT [1].

Input text	Paraphrases
it was good in spite of the taste.	despite the flavor, it felt good.
you seem to be an excellent burglar when the time comes.	when the time comes, you'll be a great thief.

Definition 2. *Given an input sentence* $\mathcal{X} = \{x_1, x_2, ..., x_m\}$ *with* m *words, the target of paraphrase generation is to generate an output sequence* $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ *with* n *words. The goal of paraphrase generation is to learn a mapping function* f*:* $\mathcal{X} \rightarrow \mathcal{Y}$*, where* f *convert input text* $\mathcal{X}$ *to output text* $\mathcal{Y}$ *that convey the same meaning but with different expressions from input. The generation process is formulated as maximizing likelihood of generating a sequence of questions* $\mathcal{Y}$ *given the input* $\mathcal{X}$*, i.e.,* $P(\mathcal{Y}|\mathcal{X}; \theta)$*.*

Early approaches to paraphrase generation are based on rules, leveraging manually crafted [61] or automatically acquired paraphrase rules [62] as their foundation. However, the rule-based approaches are not flexible to extend to various conditions for natural language structure.

Recent progress in paraphrase generation is largely due to neural models, especially sequence-to-sequence models, trained with large-scale parallel data. The first practice of adopting seq2seq architecture to resolve the task of paraphrase generation is by [63].

They utilized the LSTM model to encode the sequential text. However, the standard seq2seq generation models often suffer from exposure bias. Inspired by the early attempts, researchers have applied search-and-learning approaches for supervised paraphrasing, such as reinforcement learning (RL) [64, 65, 66] and learning-to-search (L2S) [67]. Rather than minimizing the loss function, the RL-based approaches directly optimize the reward, such as the desired evaluation metrics, perplexity score or sentence matching scores.

Researchers have proposed roundtrip translation for paraphrasing, i.e., translating a source sentence into a pivot language, and then translating it back into the original language [68, 69, 70]. Although no supervision of paraphrases is needed, the success of this approach depends on high-quality machine translation (MT) systems, hence requiring large-scale parallel MT datasets. This can be thought of as distant supervision for paraphrasing.

In the unsupervised setting, paraphrases can be generated by either a variational latent-space sampler [71] or a word-space Metropolis–Hastings (MH) sampler [72]. Another line of research is to cast the text generation into the framework of reinforcement

learning, using a pre-defined reward function to guide the training of policy for paraphrase generation [73]. By decreasing the temperature of the stationary distribution, Liu et al. [74] show that search-based formulation outperforms sampling for unsupervised text generation. Our work in Chapter 4 further extends it to the learning-from-search framework, improving both accuracy and inference efficiency.

2.2.3 Text Style Transfer

As a long-standing text generation problem, text style transfer is to alter the style attributes of text while preserving the original meaning [75], such as sentiment [17], politeness [76], formality [21], prose style [77] and aesthetic styles of poetry [78]. For instance, given an informal text like "*Ask her about that thing dude...*", the goal of style transfer is to obtain its formalized version while maintaining the original sentence's meaning, such as "*Please inquire her opinion regarding that issue.*" The formal definition of text style transfer is as follows:

Definition 3. *Given an input sentence* $\mathcal{X} = \{x_1, x_2, ..., x_m\}$ *with* m *words and a target style* $\mathbf{s}$*, the problem of text style transfer*

aims to generate an output sequence $\mathcal{X}^s = \{x_1^s, x_2^s, ..., x_n^s\}$ *with* n *words that exhibit the desired target style* $\mathbf{s}$*. The goal of text style transfer is to obtain a mapping function* f*:* $\mathcal{X} \rightarrow \mathcal{X}^s$*, where* f *converts the input text* $\mathcal{X}$ *to output text* $\mathcal{X}^s$ *adhering to the desired target style* $\mathbf{s}$*, while preserving the original meaning of the input text. The mapping function* f *is trained to capture the underlying patterns and dependencies between the input text, the desired style, and the corresponding output text, by optimizing the parameters* θ *to maximize the likelihood of generating target-styled sentences* $P(\mathcal{X}^s|\mathcal{X}, \mathbf{s}; \theta)$

Typically, existing solutions to text style transfer can be divided into three categories: **parallel supervised**, **non-parallel supervised** (with only style labels), and **purely unsupervised**.

With the parallel data, supervised style transfer trains a sequence-to-sequence model [21] for style transformation, whereas purely unsupervised style transfer replies on disentangling latent space [79].

However, the parallel data for test style transfer is often not easy to collect. Most previous work on text style transfer is in the non-parallel supervised setting, assuming only style labels are available. Variational auto-encoder (VAE) [80] and adversarial learning are

well-adopted ideas for text style transfer, which aims to disentangle the style and content of texts in latent space [17, 81, 82]. Due to the issue of computational inefficiency and unstable training, some simpler approaches propose editing partial texts of input.

Different from previous methods which assume the content and style are implicitly combined, the prototype editing methods believe the text consists of stylized words and non-stylized content words. Li et al. [20] achieves text style transfer by identifying and replacing the stylized n-grams in the source text with stylized words with target style. The infilling of words with target style can also be achieved by pretrained Transformer models. Malmi et al. [83] fine-tunes the BERT model for different styles and conducts in-place span replacement for target style through masked language modeling. Some work directly integrates the AMR representation as the intermediate style agnostic representation. [84] proposes an AMR-to-text decoder to generate sentences with target style, while [85], an editing-based approach, realize the style transfer by editing stylistic nodes in the parsed AMR graph of source text.

Constructing pseudo-parallel training corpora can provide substantial supervision signals to facilitate the training of a seq2seq

text style transfer model. Some approaches endeavor to construct pseudo-parallel training data with the label information and train a style transduction model in a supervised way [86]. While Reid and Zhong [87] adopt the pseudo parallel corpus to train a tagger model and predict token-level edit operations to guide revision.

2.2.4 Text Simplification

The task of text simplification is to revise the original text with simpler language while keeping the meaning unchanged [88]. The simplification process involves modifying the complexity of the input text by adjusting sentence structures, replacing complex words or phrases with simpler alternatives, rephrasing convoluted expressions, and adapting the overall linguistic style. Its target is to improve the readability of text.

Definition 4. *Let $\mathcal{X}$ denotes an input sentence length at m, $\mathcal{X} = \{x_1, x_2, ..., x_m\}$, the problem of text simplification aims to generate an output sequence $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ with length at n that is simpler and easier to comprehend compared to the original sentence $\mathcal{X}$. The goal of text simplification is to get a mapping function $f: \mathcal{X} \rightarrow \mathcal{Y}$, where f converts the input text $\mathcal{X}$*

to output text $\mathcal{Y}$ while preserving the core meaning and reducing its complexity. The mapping function f, parameterized by θ, is trained to capture the underlying patterns and dependencies between the input text and the corresponding simplified output text to maximize the likelihood of generating simplified sentences $P(\mathcal{Y}|\mathcal{X};\theta)$.

Early approaches to text simplification primarily relied on rule-based methods. The first effort towards automated simplification is a grammar and style checker developed for writers of simplified English [89]. Lexical substitution techniques were employed to replace complex words and phrases with simpler alternatives [90], while sentence splitting and syntactic transformations were applied to break down complex sentences into simpler ones [91, 92]. These rule-based methods, though effective to some extent, often lacked the ability to handle complex linguistic phenomena and adapt to diverse contexts.

Recent approaches to automatic text simplification adopt neural methods. Wang et al. [93] proposed using an RNN-based neural machine translation model (NMT) for the task of text simplification. Nevertheless, they highlighted the absence of aligned pairs of complex and simple sentences for the development of such a

model and reported the efforts solely on applying NMT on lexical simplification. Xu et al. [94] pioneeringly proposed a large complex-to-simple parallel corpus, making it possible to address text simplification at sentence level using neural networks. Zhang and Lapata [6] cast simplification into the framework of reinforcement learning with three types of reward: simplicity, relevance and fluency. Dong et al. [95] suggests explicitly modeling the edit operations. Previous studies explicitly define three simplification components: sentence splitting, deletion and paraphrasing [96, 97]. Based on these components, Maddela et al. [98] proposes a pipeline, where the first part focuses on syntactic simplification, while the second part focuses on lexical and phrasal simplification.

There are also some efforts for unsupervised text simplification. Some early attempts are built upon the heuristics of the task itself. For instance, Narayan and Gardent [99] propose a task-specific pipeline for sentence simplification. The first sequence-to-sequence framework is proposed by [100], who employ a shared encoder and multiple decoders which learns to control the simplicity of generated text via the adversarial loss. To improve the interpretability, Kumar et al. [101] design an iterative, edit-based approach. Specif-

ically, they parse the sentence to a constituency tree, conditioned on which they conduct syntactic simplification.

2.3 Evaluation Metrics

This section introduces the evaluation methods of text generation. According to the assessing criteria, existing evaluation approaches can be categorized into two types: **automatic evaluation** and **human evaluation**. While automatic metrics provide objective measures for evaluating the generated output, human evaluation offers valuable insights into subjective aspects such as fluency, coherence, and overall quality of the generated text. For each category, we elaborate on the principles of the relevant evaluation metrics and the corresponding natural language generation tasks.

2.3.1 Automatic Evaluation

With the increasing attention on the NLG research and rapid growth of benchmark datasets, a fast, cross-platform and generalized automatic evaluation paradigm for text generation becomes quite important. These standardized automatic evaluation metrics serve as a judge to compare the performance between proposed

new solutions and the state-of-the-art approach, automatically maintaining a fair competition system.

The major issue in NLG is to determine the accuracy of the predicted text. To measure the accuracy of the generated text, people usually compute the similarity between the prediction and the human-written references. Specifically, the n-gram overlap between predicted text and ground-truth text is computed. Existing well-adopted automatic metrics for calculating accuracy include BLEU (Bilingual Evaluation Understudy) [102], ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [103] and METEOR ((Metric for Evaluation of Translation with Explicit ORdering) [104]. These metrics provide automated means to assess the fluency, adequacy, grammaticality, and overall quality of the generated text, enabling researchers and developers to evaluate and compare different NLG systems objectively.

BLEU, the Bilingual Evaluation Understudy, computes a similarity score between the machine-generated sentence and one or more human-written references. The similarity score is based on n-gram precision and a penalty term, where $n \in \{1, 2, 3, 4\}$. Despite its simplicity, BLEU has been widely adopted in NLG

evaluations due to its ease of use and ability to provide a quick assessment of the generated text's similarity to the desired reference. Although it is originally proposed for automatic evaluation in machine translation, it has been broadly applied in the evaluation of text generation tasks.

ROUGE, the Recall-Oriented Understudy for Gisting Evaluation, computes the n-gram recall between the generated text and a set of human references. It is originally proposed for the evaluation of long text in text summarization. ROUGE offers a comprehensive evaluation by considering multiple levels of textual units and providing a quantitative measure of the similarity and informativeness of the generated output.

METEOR, the Metric for Evaluation of Translation with Explicit ORdering, computes the harmonic mean of the unigram precision and recall to compute a score that reflects the overall semantic similarity and fluency. METEOR takes into account variations in word choice, sentence structure, stemming, and synonymy, allowing for a more nuanced evaluation. By considering both exact and paraphrased matches, METEOR provides a comprehensive assessment that captures both lexical and semantic

aspects, making it a valuable tool for evaluating the effectiveness and adequacy of NLG systems.

CIDEr [105], Consensus-based Image Description Evaluation, focuses on capturing consensus in human judgments through measuring n-gram co-occurrence. CIDEr goes beyond lexical similarity and emphasizes capturing the consensus in human judgments. It measures the co-occurrence of n-grams between the generated descriptions and multiple human reference descriptions. CIDEr is particularly valuable in NLG tasks related to image captioning, where the generated descriptions should be not only accurate but also diverse and capturing the essence of the visual content.

With the emergence of large-scale pretrained models [44, 46, 47, 50], there has been a large amount of research work focusing on utilizing the expressive power of contextualized word embeddings to improve the evaluation of NLG tasks.

BERTScore [106]. Different from the exact matching strategy in previous metrics, BERTScore computes the cosine similarity score for each word in the candidate sentence with each one in the references sentence, using the contextual embeddings from BERT. This metric is proven to possess a high correlation with human

judgments in both human-level and system-level evaluations.

BLEURT [107] is a regression-based evaluation metric, where the authors fine-tune a BERT checkpoint to obtain an evaluation model for various NLG tasks. This evaluation model is trained to accurately predict scores similar to human judgments. BLEURT has shown promising results in capturing the nuances of NLG output, providing a more robust evaluation metric that aligns with human judgments.

BARTScore [108] formulates the evaluation of generated text as a sequence-to-sequence generation task with a pre-trained model, BART. Concretely speaking, BARTScore calculates the weighted log probability of one text given another. The weights are introduced to distinguish the emphasis on different tokens. BARTScore measures the generation quality from four dimensions: *faithfulness* (from source document to hypothesis), *precision* (from reference text to output text), *recall* (from output text to reference text) and *F-score* (the arithmetic average of *precistion* and *recall*).

In addition to the generation quality, there are attribute-specific evaluation metrics for the controllable text generation tasks. In paraphrase generation, the control factor is the lexical dissimilarity

and semantic equivalence between input and output text. ***i*BLEU** is proposed to evaluate the quality of paraphrased text by jointly considering the semantic adequacy and textual dissimilarity [109]. In the context of text simplification, **SARI** is the first automatic metric, which calculates the arithmetic average of n-gram precision and recall of three simplification operations: delete, keep and add [94]. In terms of text style transfer, a style classifier is employed to compute the score to which the predicted text confronts the expected style.

2.3.2 Human Evaluation

The ultimate goal of text generation is to generate text that is indistinguishable from the human-written counterpart. Given that automatic evaluation metrics cannot fully mimic human judgment o fall short in capturing the nuanced aspects of language generation that are vital for user satisfaction [110, 111], human evaluation becomes a crucial component of experimental analysis in text generation tasks. By incorporating human judgment, NLG systems can be refined to meet user expectations, ensuring that the generated text is not only linguistically accurate but also compelling,

engaging, and tailored to the intended audience.

Researchers define specific criteria and guidelines for assessing the quality of the generated text. Generally, the quality assessment primarily focuses on the following dimensions: *coherence*, *correctness* and *naturalness*. *Coherence* refers to the consistency between input and output text, while *correctness* assesses the absence of hallucinated information and grammatical errors in the generated content. *Naturalness* evaluates the extent to which the generated text resembles human-written text. Apart from the textual quality, other domain-specific criteria are relevant to the application.

A diverse group of human evaluators is chosen to represent the target audience or desired expertise. Evaluators should have a good understanding of the task and context, and they may be experts in the subject matter or general users. Human evaluators are presented with samples from different systems, including one or more baseline models and the proposed approach. They are asked to score the samples based on their quality.

Evaluators assess the generated text based on the predefined criteria. They may rate the text on a numerical scale, provide rankings, or offer qualitative feedback. Pairwise comparison or

ranking methods can also be employed to compare different generated outputs. Two commonly used scoring schemes include the 5-point Likert scale and direct rating within predefined rating scales.

However, when multiple evaluators are involved, the validity of the evaluations is contingent on their consistency in scoring the samples. To determine the level of agreement between different evaluators, inter-annotator agreement scores are computed, such as Cohen's kappa [112] or Fleiss' kappa [113]. Higher agreement scores among the evaluators indicate that the samples are distinguishable and the evaluators are consistent in assessing the quality of the generated text, leading to more reliable results.

□ **End of chapter.**

Chapter 3

Improving Semantic Fidelity with To the Point Context

In this chapter, we investigate the method to improve semantic fidelity in the context of question generation tasks and proposed a novel pipeline to improve the semantic accuracy of generated questions with to the point context. The introduction of this research problem is presented in Section 3.1, followed by the problem definition in Section 3.2. In Section 3.3, we describe the approach for extracting answer-relevant relations and the structure of the proposed end-to-end generation framework. Experimental setups are presented in Section 3.4, with results reported in Section 3.5 In Section 3.6, we provide a summary of the main findings and

contributions of this chapter.

3.1 Introduction

Question Generation (QG) is the task of automatically creating questions from a range of inputs, such as natural language text [54], knowledge base [59] and image [114]. QG is an increasingly important area in NLP with various application scenarios such as intelligence tutor systems, open-domain chatbots and question answering dataset construction. In this chapter, we focus on question generation from reading comprehension materials like SQuAD [115]. As shown in Figure 3.1, given a sentence in the reading comprehension noindent**a**nd the text fragment (i.e., the answer) that we want to ask about, we aim to generate a question that is asked about the specified answer.

Question generation for reading comprehension is firstly formalized as a declarative-to-interrogative sentence transformation problem with predefined rules or templates [52, 54]. With the rise of neural models, Du et al. [2] propose to model this task under the sequence-to-sequence (seq2seq) learning framework [35] with attention mechanism [116]. However, question generation is

Sentence: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month.
Reference Question: What is New York City 's daily January mean temperature in degrees celsius ?
Baseline Prediction: What is the coldest temperature in Celsius ?
Structured Answer-relevant Relation: (The daily mean temperature in January; is; 32.6 °F (0.3 °C))

Figure 3.1: An example SQuAD question with the baseline's prediction. The answer ("0.3 °C") is highlighted.

a one-to-many sequence generation problem, i.e., several aspects can be asked given a sentence. Zhou et al. [3] propose the answer-aware question generation setting which assumes the answer, a contiguous span inside the input sentence, is already known before question generation. To capture answer-relevant words in the sentence, they adopt a BIO tagging scheme to incorporate the answer position embedding in seq2seq learning. Furthermore, Sun et al. [117] propose that tokens close to the answer fragments are more likely to be answer-relevant. Therefore, they explicitly encode the relative distance between sentence words and the answer via position embedding and position-aware attention.

Although existing proximity-based answer-aware approaches achieve reasonable performance, we argue that such intuition may

Table 3.1: Performance for the average relative distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question. (Bn: BLEU-n, MET: METEOR, R-L: ROUGE-L)

Distance		B1	B2	B3	B4	MET	R-L
0∼10	(72.8% of #)	45.25	30.31	22.06	16.54	21.54	46.26
>10	(27.2% of #)	35.67	21.72	14.82	10.46	16.72	37.63

not apply to all cases especially for sentences with complex structure. For example, Figure 3.1 shows such an example where those approaches fail. This sentence contains a few facts and due to the parenthesis (i.e. "the area's coldest month"), some facts intertwine: "The daily mean temperature in January is 0.3°C" and "January is the area's coldest month". From the question generated by a proximity-based answer-aware baseline, we find that it wrongly uses the word "coldest" but misses the correct word "mean" because "coldest" has a shorter distance to the answer "0.3°C".

In summary, their intuition that "the neighboring words of the answer are more likely to be answer-relevant and have a higher chance to be used in the question" is not reliable. To quantitatively show this drawback of these models, we implement the approach proposed by Sun et al. [117] and analyze its performance under different relative distances between the answer and other non-stop sentence words that also appear in the ground truth question. The

results are shown in Table 3.1. We find that the performance drops at most 36% when the relative distance increases from "$0 \sim 10$" to "> 10". In other words, when the useful context is located far away from the answer, current proximity-based answer-aware approaches will become less effective, since they overly emphasize neighboring words of the answer.

To address this issue, we extract the structured answer-relevant relations from sentences and propose a method to jointly model such structured relation and the unstructured sentence for question generation. The structured answer-relevant relation is likely to be to the point context and thus can help keep the generated question to the point. For example, Figure 3.1 shows our framework can extract the right answer-relevant relation ("The daily mean temperature in January", "is", "32.6°F (0.3°C)") among multiple facts. With the help of such structured information, our model is less likely to be confused by sentences with a complex structure. Specifically, we firstly extract multiple relations with an off-the-shelf Open Information Extraction (OpenIE) toolbox [118], then we select the relation that is most relevant to the answer with carefully designed heuristic rules.

Nevertheless, it is challenging to train a model to effectively utilize both the unstructured sentence and the structured answer-relevant relation because both of them could be noisy: the unstructured sentence may contain multiple facts which are irrelevant to the target question, while the limitation of the OpenIE tool may produce less accurate extracted relations. To explore their advantages simultaneously and avoid the drawbacks, we design a gated attention mechanism and a dual copy mechanism based on the encoder-decoder framework, where the former learns to control the information flow between the unstructured and structured inputs, while the latter learns to copy words from two sources to maintain the informativeness and faithfulness of generated questions.

In the evaluations on the SQuAD dataset, our system achieves significant and consistent improvement as compared to all baseline methods. In particular, we demonstrate that the improvement is more significant with a larger relative distance between the answer and other non-stop sentence words that also appear in the ground truth question. Furthermore, our model is capable of generating diverse questions for a single sentence-answer pair where the sentence conveys multiple relations of its answer fragment.

3.2 Problem Definition

We formalize our task as an answer-aware Question Generation (QG) problem [119], which assumes answer phrases are given before generating questions. Moreover, answer phrases are shown as text fragments in passages. Formally, given the sentence S, the answer A, and the answer-relevant relation M, the task of QG aims to find the best question $\overline{Q}$ such that,

$$\overline{Q} = \arg\max_{Q} \text{Prob}(Q|S, A, M), \tag{3.1}$$

where A is a contiguous span inside S.

3.3 Methodology

In this section, we first introduce the task definition and our protocol to extract structured answer-relevant relations. Then we formalize the task under the encoder-decoder framework with gated attention and dual copy mechanism.

Sentence: Beyoncé received critical acclaim and commercial success, selling one million digital copies worldwide in six days; The New York Times noted the album's unconventional, unexpected release as significant.
N-ary Relations:
0.85 (Beyoncé; received commercial success selling; one million digital copies worldwide; in six days)
0.92 (The New York Times; noted; the album's unconventional, unexpected release as significant)

Sentence: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month.
N-ary Relations:
0.95 (The daily mean temperature in January; is; 32.6 °F (0.3 °C))
0.94 (temperatures; drop; to 10 °F (12 °C); several times per winter; usually)
0.90 (temperatures; reach; 50 °F)

Figure 3.2: Examples for n-ary extractions from sentences using OpenIE. Confidence scores are shown at the beginning of each relation. Answers are highlighted in sentences. Waved relations are selected according to our criteria in Section 3.3.1.

3.3.1 Answer-relevant Relation Extraction

We utilize an off-the-shelf toolbox of OpenIE [1] to the derive structured answer-relevant relations from sentences as to the point contexts. Relations extracted by OpenIE can be represented either in a triple format or in an n-ary format with several secondary arguments, and we employ the latter to keep the extractions as informative as possible and avoid extracting too many similar

[1] http://openie.allenai.org/

relations in different granularities from one sentence. We join all arguments in the extracted n-ary relation into a sequence as our to the point context. Figure 3.3 shows n-ary relations extracted from OpenIE. As we can see, OpenIE extracts multiple relations for complex sentences. Here we select the most informative relation according to three criteria in the order of descending importance: (1) having the maximal number of overlapped tokens between the answer and the relation; (2) being assigned the highest confidence score by OpenIE; (3) containing maximum non-stop words. As shown in Figure 3.3, our criteria can select answer-relevant relations (waved in Figure 3.3), which is especially useful for sentences with extraneous information. In rare cases, OpenIE cannot extract any relation, we treat the sentence itself as the to the point context.

Table 3.2 shows some statistics to verify the intuition that the extracted relations can serve as more to the point context. We find that the tokens in relations are 61% more likely to be used in the target question than the tokens in sentences, and thus they are more to the point. On the other hand, on average the sentences contain one more question token than the relations (1.86 v.s. 2.87). Therefore, it is still necessary to take the original sentence into

Table 3.2: Comparisons between sentences and answer-relevant relations. Overlapped words are those non-stop tokens co-occurring in the source (sentence/relation) and the target question. Copy ratio means the proportion of source tokens that are used in the question.

	Sentence	Answer-relevant Relation
Avg. length	32.46	13.04
# overlapped words	2.87	1.86
Copy ratio	8.85%	14.26%

account to generate a more accurate question.

3.3.2 Proposed Framework

Overview. As shown in Figure 3.3, our framework consists of four components (1) Sentence Encoder and Relation Encoder, (2) Decoder, (3) Gated Attention Mechanism and (4) Dual Copy Mechanism. The sentence encoder and relation encoder encode the unstructured sentence and the structured answer-relevant relation, respectively. To select and combine the source information from the two encoders, a gated attention mechanism is employed to jointly attend to both contextualized information sources, and a dual copy mechanism copies words from either the sentence or the relation.

Answer-aware Encoder. We employ two encoders to integrate information from the unstructured sentence S and the answer-

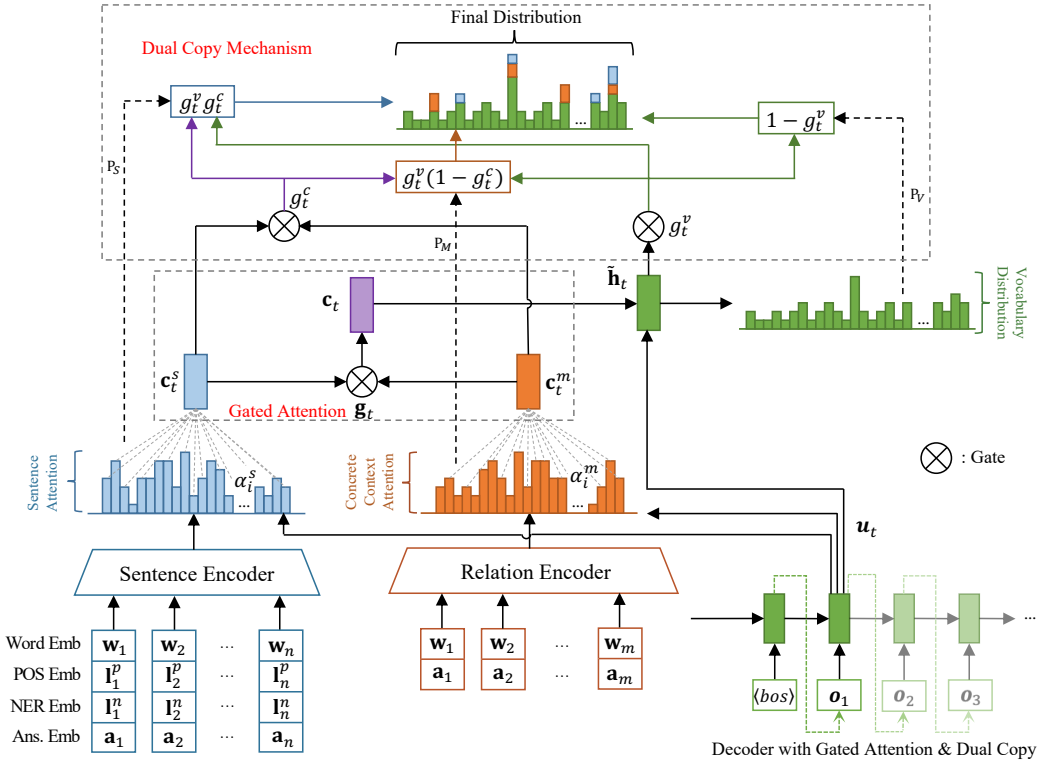

Figure 3.3: The framework of our proposed model. *(Best viewed in color)*

relevant relation M separately. Sentence encoder takes in feature-enriched embeddings including word embeddings $\mathbf{w}$, linguistic embeddings $\mathbf{l}$ and answer position embeddings $\mathbf{a}$. We follow [3] to transform POS and NER tags into continuous representation ($\mathbf{l}^p$ and $\mathbf{l}^n$) and adopt a BIO labeling scheme to derive the answer position embedding (B: the first token of the answer, I: tokens within the answer fragment except the first one, O: tokens outside of the answer fragment). For each word w_i in the sentence S, we simply concatenate all features as input: $\mathbf{x}_i^s = [\mathbf{w}_i; \mathbf{l}_i^p; \mathbf{l}_i^n; \mathbf{a}_i]$. Here

$[\mathbf{a}; \mathbf{b}]$ denotes the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$.

We use bidirectional LSTMs to encode the sentence $(\mathbf{x}_1^s, \mathbf{x}_2^s, ..., \mathbf{x}_n^s)$ to get a contextualized representation for each token:

$$\overrightarrow{\mathbf{h}}_i^s = \overrightarrow{\text{LSTM}}(\overrightarrow{\mathbf{h}}_{i-1}^s, \mathbf{x}_i^s), \ \overleftarrow{\mathbf{h}}_i^s = \overleftarrow{\text{LSTM}}(\overleftarrow{\mathbf{h}}_{i+1}^s, \mathbf{x}_i^s),$$

where $\overrightarrow{\mathbf{h}}_i^s$ and $\overleftarrow{\mathbf{h}}_i^s$ are the hidden states at the i-th time step of the forward and the backward LSTMs. The output state of the sentence encoder is the concatenation of forward and backward hidden states: $\mathbf{h}_i^s = [\overrightarrow{\mathbf{h}}_i^s; \overleftarrow{\mathbf{h}}_i^s]$. The contextualized representation of the sentence is $(\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_n^s)$.

For the relation encoder, we firstly join all items in the n-ary relation M into a sequence. Then we only take answer position embedding as an extra feature for the sequence: $\mathbf{x}_i^m = [\mathbf{w}_i; \mathbf{a}_i]$. Similarly, we take another bidirectional LSTM to encode the relation sequence and derive the corresponding contextualized representation $(\mathbf{h}_1^m, \mathbf{h}_2^m, ..., \mathbf{h}_n^m)$.

Decoder. We use an LSTM as the decoder to generate the question. The decoder predicts the word probability distribution at each decoding timestep to generate the question. At the t-th timestep, it reads the word embedding $\mathbf{w}_t$ and the hidden state

$\mathbf{u}_{t-1}$ of the previous timestep to generate the current hidden state:

$$\mathbf{u}_t = \text{LSTM}(\mathbf{u}_{t-1}, \mathbf{w}_t). \tag{3.2}$$

Gated Attention Mechanism. We design a gated attention mechanism to jointly attend to the sentence representation and the relation representation. For sentence representation $(\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_n^s)$, we employ the Luong et al. [116]'s attention mechanism to obtain the sentence context vector $\mathbf{c}_t^s$,

$$a_{t,i}^s = \frac{\exp(\mathbf{u}_t^\top \mathbf{W}_a \mathbf{h}_i^s)}{\sum_j \exp(\mathbf{u}_t^\top \mathbf{W}_a \mathbf{h}_j^s)}, \ \mathbf{c}_t^s = \sum_i a_{t,i}^s \mathbf{h}_i^s,$$

where $\mathbf{W}_a$ is a trainable weight. Similarly, we obtain the vector $\mathbf{c}_t^m$ from the relation representation $(\mathbf{h}_1^m, \mathbf{h}_2^m, ..., \mathbf{h}_n^m)$. To jointly model the sentence and the relation, a gating mechanism is designed to control the information flow from two sources:

$$\mathbf{g}_t = \text{sigmoid}(\mathbf{W}_g[\mathbf{c}_t^s; \mathbf{c}_t^m]), \tag{3.3}$$

$$\mathbf{c}_t = \mathbf{g}_t \odot \mathbf{c}_t^s + (\mathbf{1} - \mathbf{g}_t) \odot \mathbf{c}_t^m, \tag{3.4}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h[\mathbf{u}_t; \mathbf{c}_t]), \tag{3.5}$$

where $\odot$ represents element-wise dot production and $\mathbf{W}_g, \mathbf{W}_h$ are

trainable weights. Finally, the predicted probability distribution over the vocabulary V is computed as:

$$\mathrm{P}_V = \mathrm{softmax}(\mathbf{W}_V \tilde{\mathbf{h}}_t + \mathbf{b}_V), \tag{3.6}$$

where $\mathbf{W}_V$ and $\mathbf{b}_V$ are parameters.

Dual Copy Mechanism. To deal with the rare and unknown words, the decoder applies the pointing method [120, 121, 122] to allow copying a token from the input sentence at the t-th decoding step. We reuse the attention score α_t^s and α_t^m to derive the copy probability over two source inputs:

$$\mathrm{P}_S(w) = \sum_{i:w_i=w} \alpha_{t,i}^s, \mathrm{P}_M(w) = \sum_{i:w_i=w} \alpha_{t,i}^m.$$

Different from the standard pointing method, we design a dual copy mechanism to copy from two sources with two gates. The first gate is designed for determining copy tokens from two sources of inputs or generating next word from P_V, which is computed as $g_t^v = \mathrm{sigmoid}(\mathbf{w}_g^v \tilde{\mathbf{h}}_t + b_g^v)$. The second gate takes charge of selecting the source (sentence or relation) to copy from, which is computed as $g_t^c = \mathrm{sigmoid}(\mathbf{w}_g^c [\mathbf{c}_t^s; \mathbf{c}_t^m] + b_g^c)$. Finally, we combine

all probabilities P_V, P_S and P_M through two soft gates g_t^v and g_t^c. The probability of predicting w as the t-th token of the question is:

$$\mathrm{P}(w) = (1 - g_t^v)\mathrm{P}_V(w) + g_t^v g_t^c \mathrm{P}_S(w) + g_t^v(1 - g_t^c)\mathrm{P}_M(w).$$

Training and Inference. Given the answer A, sentence S and relation M, the training objective is to minimize the negative log-likelihood with regard to all parameters:

$$\mathcal{L} = -\sum_{Q \in \{Q\}} log\ \mathrm{P}(Q|A, S, M; \theta), \tag{3.7}$$

where $\{Q\}$ is the set of all training instances, θ denotes model parameters and $\log P(Q|A, S, M; \theta)$ is the conditional log-likelihood of Q.

In testing, our model targets to generate a question Q by maximizing:

$$\overline{Q} = \arg\max_{Q} log\ \mathrm{P}(Q|A, S, M; \theta). \tag{3.8}$$

Table 3.3: Dataset statistics on Du Split [2] and Zhou Split [3].

	Du Split	Zhou Split
# pairs (Train)	74689	86635
# pairs (Dev)	10427	8965
# pairs (Test)	11609	8964
Sentence avg. tokens	32.56	32.72
Question avg. tokens	11.42	11.31

3.4 Experimental Setting

3.4.1 Dataset and Metrics

We conduct experiments on the SQuAD dataset [115]. It contains 536 Wikipedia articles and 100k crowd-sourced question-answer pairs. The questions are written by crowd-workers and the answers are spans of tokens in the articles. We employ two different data splits by following Zhou et al. [3] [2] and Du et al. [2] [3]. In Zhou et al. [3], the original SQuAD development set is evenly divided into dev and test sets, while Du et al. [2] treats SQuAD development set as its development set and splits original SQuAD training set into a training set and a test set. We also filter out questions which do not have any overlapped non-stop words with the corresponding sentences and perform some preprocessing steps, such as tokenization and sentence splitting. The data statistics

[2]https://res.qyzhou.me/redistribute.zip
[3]https://github.com/xinyadu/nqg/tree/master/data/raw

Table 3.4: The main experimental results for our model and several baselines in Du Split [2] version of SQuAD. '-' means no results reported in their papers. (Bn: BLEU-n, MET: METEOR, R-L: ROUGE-L)

	Du Split [2]					
	B1	B2	B3	B4	MET	R-L
s2s [2]	43.09	25.96	17.50	12.28	16.62	39.75
NQG++ [3]	-	-	-	-	-	-
M2S+cp [8]	-	-	-	13.98	18.77	42.72
s2s+MP+GSA [119]	43.47	28.23	20.40	15.32	19.29	43.91
Hybrid model [117]	-	-	-	-	-	-
ASs2s [56]	-	-	-	16.20	19.92	43.96
Our model	**45.66**	**30.21**	**21.82**	**16.27**	**20.36**	**44.35**

Table 3.5: The main experimental results for our model and several baselines in Zhou Split [3] version of SQuAD. '-' means no results reported in their papers. (Bn: BLEU-n, MET: METEOR, R-L: ROUGE-L)

	Zhou Split [3]					
	B1	B2	B3	B4	MET	R-L
s2s [2]	-	-	-	-	-	-
NQG++ [3]	-	-	-	13.29	-	-
M2S+cp [8]	-	-	-	13.91	-	-
s2s+MP+GSA [119]	**44.51**	29.07	21.06	15.82	19.67	44.24
Hybrid model [117]	43.02	28.14	20.51	15.64	-	-
ASs2s [56]	-	-	-	16.17	-	-
Our model	44.40	**29.48**	**21.54**	**16.37**	**20.68**	**44.73**

are given in Table 3.3.

We evaluate with all commonly-used metrics in question generation [2]: BLEU-1 (B1), BLEU-2 (B2), BLEU-3 (B3), BLEU-4 (B4) [102], METEOR (MET) [123] and ROUGE-L (R-L) [103]. We use the evaluation script released by Chen et al. [124].

3.4.2 Baseline Methods

We compare with the following models.

- s2s [2] proposes an attention-based sequence-to-sequence neural network for question generation.
- NQG++ [3] takes the answer position feature and linguistic features into consideration and equips the seq2seq model with copy mechanism.
- M2S+cp [8] conducts multi-perspective matching between the answer and the sentence to derive an answer-aware sentence representation for question generation.
- s2s+MP+GSA [119] introduces a gated self-attention into the encoder and a maxout pointer mechanism into the decoder. We report their sentence-level results for a fair comparison.
- Hybrid [117] is a hybrid model which considers the answer embedding for the question word generation and the position of context words for modeling the relative distance between the context words and the answer.
- ASs2s [56] replaces the answer in the sentence with a special token to avoid its appearance in the generated questions.

3.4.3 Implementation Details

We take the most frequent 20k words as our vocabulary and use the GloVe word embeddings [125] for initialization. The embedding dimensions for POS, NER, answer position are set to 20. We use two-layer LSTMs in both encoder and decoder, and the LSTMs hidden unit size is set to 600.

We use dropout [126] with the probability $p = 0.3$. All trainable parameters, except word embeddings, are randomly initialized with the Xavier uniform in $(-0.1, 0.1)$ [127]. For optimization in the training, we use SGD as the optimizer with a minibatch size of 64 and an initial learning rate of 1.0. We train the model for 15 epochs and start halving the learning rate after the 8th epoch. We set the gradient norm upper bound to 3 during the training.

We adopt the teacher-forcing for the training. In the testing, we select the model with the lowest perplexity and beam search with size 3 is employed for generating questions. All hyper-parameters and models are selected on the validation dataset.

Table 3.6: Performance for the average relative distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question (BLEU is the average over BLEU-1 to BLEU-4). Values in parenthesis are the improvement percentage of Our Model over Hybrid. (a) is based on all sentences while (b) only considers long sentences with more than 20 words.

a Evaluation results on all sentences.

		Hybrid			Our Model		
		BLEU	MET	R-L	BLEU	MET	R-L
0∼10	(72.8% of #)	28.54	21.54	46.26	29.73 (4.17%)	22.03 (2.27%)	46.85 (1.28%)
>10	(27.2% of #)	20.67	16.72	37.63	22.12 (7.01%)	17.46 (4.43%)	38.47 (2.23%)

b Evaluation results on sentences with more than 20 words.

		Hybrid			Our Model		
		BLEU	MET	R-L	BLEU	MET	R-L
0∼10	(58.3% of #)	28.00	21.03	45.37	29.11 (3.96%)	21.50 (2.21%)	45.97 (1.31%)
>10	(26.6% of #)	20.58	16.66	37.53	22.04 (7.09%)	17.38 (4.30%)	38.37 (2.24%)

3.5 Results and Analysis

3.5.1 Overall Performance

Table 3.4 and 3.5 show automatic evaluation results for our model and baselines (copied from their papers). Our proposed model which combines structured answer-relevant relations and unstructured sentences achieves significant improvements over proximity-based answer-aware models [3, 117] on both dataset splits. Presumably, our structured answer-relevant relation is a generalization of the context explored by the proximity-based methods because they can only capture short dependencies around answer fragments

while our extractions can capture both short and long dependencies given the answer fragments. Moreover, our proposed framework is a general one to jointly leverage structured relations and unstructured sentences. All compared baseline models which only consider unstructured sentences can be further enhanced under our framework.

Recall that existing proximity-based answer-aware models perform poorly when the distance between the answer fragment and other non-stop sentence words that also appear in the ground truth question is large (Table 3.1). Here we investigate whether our proposed model using the structured answer-relevant relations can alleviate this issue or not, by conducting experiments for our model under the same setting as in Table 3.1. The broken-down performances by different relative distances are shown in Table 3.6a. We find that our proposed model outperforms Hybrid (our re-implemented version for this experiment) on all ranges of relative distances, which shows that the structured answer-relevant relations can capture both short and long term answer-relevant dependencies of the answer in sentences. Furthermore, comparing the performance difference between Hybrid and our model, we

find the improvements become more significant when the distance increases from "$0 \sim 10$" to "> 10". One reason is that our model can extract relations with distant dependencies to the answer, which greatly helps our model ignore the extraneous information. Proximity-based answer-aware models may overly emphasize the neighboring words of answers and become less effective as the useful context becomes further away from the answer in the complex sentences. In fact, the breakdown intervals in Table 3.6a naturally bound its sentence length, say for "> 10", the sentences in this group must be longer than 10. Thus, the length variances in these two intervals could be significant. To further validate whether our model can extract long term dependency words. We rerun the analysis of Table 3.6b only for long sentences (length > 20) of each interval. The improvement percentages over Hybrid are shown in Table 3.6b, which become more significant when the distance increases from "$0 \sim 10$" to "> 10".

3.5.2 Case Study

Figure 3.4 provides example questions generated by crowd-workers (ground truth questions), the baseline Hybrid [117], and our model.

Sentence: Beyoncé received critical acclaim and commercial success, selling one million digital copies worldwide in six days; The New York Times noted the album 's unconventional, unexpected release as significant. **Reference Question**: How many digital copies of her fifth album did Beyoncé sell in six days? **Baseline Prediction**: How many digital copies did the New York Times sell in six days ? **Structured Answer-relevant Relation**: (Beyoncé; received commercial success selling; one million digital copies worldwide; in six days) **Our Model Prediction**: How many digital copies did Beyoncé sell in six days ?
Sentence: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month. **Reference Question**: What is New York City 's daily January mean temperature in degrees celsius ? **Baseline Prediction**: What is the coldest temperature in Celsius ? **Structured Answer-relevant Relation**: (The daily mean temperature in January; is; 32.6 °F (0.3 °C)) **Our Model Prediction**: In degrees Celsius , what is the average temperature in January ?

Figure 3.4: Example questions (with answers highlighted) generated by crowdworkers (ground truth questions), the baseline model and our model.

In the first case, there are two subsequences in the input and the answer has no relation with the second subsequence[4]. However, we see that the baseline model prediction copies irrelevant words "The New York Times" while our model can avoid using the extraneous subsequence "The New York Times noted ..." with the help of the structured answer-relevant relation. Compared with

[4]One might think that the two subsequences should be regarded as individual sentences, however, several off-the-shelf tools do recognize them as one sentence.

the ground truth question, our model cannot capture the cross-sentence information like "her fifth album", where the techniques in noindent-level QG models [119] may help. In the second case, as discussed in Section 3.1, this sentence contains a few facts and some facts intertwine. We find that our model can capture distant answer-relevant dependencies such as "mean temperature" while the proximity-based baseline model wrongly takes neighboring words of the answer like "coldest" in the generated question.

3.5.3 Diverse Question Generation

Another interesting observation is that for the same answer-sentence pair, our model can generate diverse questions by taking different answer-relevant relations as input. Such capability improves the interpretability of our model because the model is given not only what to be asked (i.e., the answer) but also the related fact (i.e., the answer-relevant relation) to be covered in the question. In contrast, proximity-based answer-aware models can only generate one question given the sentence-answer pair regardless of how many answer-relevant relations in the sentence. We think such capability can also validate our motivation: questions should be generated ac-

Sentence: In July 1960, NASA Deputy Administrator Hugh L. Dryden announced the Apollo program to industry representatives at a series of Space Task Group conferences. **Relation 1**: (Hugh L. Dryden; [is] Deputy Administrator [of]; NASA) **Question 1**: Who was the NASA Deputy Administrator in 1960 ? **Relation 2**: (NASA Deputy Administrator Hugh L. Dryden; announced; the Apollo program to industry representatives at a series of Space Task Group conferences; In July 1960) **Question 2**: Who announced the Apollo program to industry representatives ?
Sentence: One of the network's strike-replacement programs during that time was the game show Duel, which premiered in December 2007. **Relation 1**: (the game show Duel; premiered; in December 2007) **Question 1**: What game premiered in December 2007 ? **Relation 2**: (One of the network's strike-replacement programs during that time; was; the game show Duel) **Question 2**: What was the name of an network 's strike - replacement programs ?

Figure 3.5: Example diverse questions (with answers highlighted) generated by our model with different answer-relevant relations.

cording to the answer-aware relations instead of neighboring words of answer fragments. Figure 3.5 show two examples of diverse question generation. In the first case, the answer fragment 'Hugh L. Dryden' is the appositive to 'NASA Deputy Administrator' but the subject to the following tokens 'announced the Apollo program ...'. Our framework can extract these two answer-relevant relations, and by feeding them to our model separately, we can receive two questions asking different relations with regard to the answer.

3.6 Summary

In this chapter, we propose a question generation system which combines unstructured sentences and structured answer-relevant relations for generation. The unstructured sentences maintain the informativeness of generated questions while structured answer-relevant relations keep the faithfulness of questions. Extensive experiments demonstrate that our proposed model achieves state-of-the-art performance across several metrics. Furthermore, our model can generate diverse questions with different structured answer-relevant relations. For future work, there are some interesting dimensions to explore, such as difficulty levels [128], noindent-level information [119] and conversational question generation [129].

□ **End of chapter.**

Chapter 4

Unsupervised Text Generation by Learning from Search

In this chapter, we present our study on unsupervised controllable text generation. In contrast to previous problem setting, we consider the data-scarce setting where no parallel training corpus is available. In Section 4.1, the background knowledge and research progress of unsupervised text generation are provided. Subsequently, we proposed an unsupervised solution by search and learning. Details of our proposed framework are covered in Section 4.2, including our search algorithm and the two learning strategies. The configuration of experiments is presented in Section 4.3. Given the empirical results, we further conduct detailed

analysis in Section 4.4. To conclude, the contents in this chapter are summarized in Section 4.5.

4.1 Introduction

Text generation refers to a wide range of tasks involving generating natural language, including machine translation [130, 131, 132], sentence simplification [99, 133], and text summarization [134, 135]. Recent success of neural text generation relies heavily on large parallel data 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 [136, 137]. 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 [130].

Recently, researchers have developed search-based techniques for unsupervised text generation [72, 74, 101, 138], 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 [72, 74], sentence summarization [138], and text simplification [101].

However, the above 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 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 a 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: 1) the uni-directional GPT2 [45], which is suitable for likelihood-based fluency evaluation and conditional generation; and 2) the bi-directional RoBERTa [47], which is better at semantic evaluation and contextual word-level prediction.

The main contributions include: 1) We propose TGLS, a prin-

cipled framework for unsupervised text generation; TGLS can be applied to different tasks if the output resembles the input and can be roughly estimated by a heuristically defined scoring function. 2) We successfully incorporate 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 [67] 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.

4.2 Methodology

Our TGLS framework involves two stages of search and learning. In the first stage, we perform simulated annealing (SA) search [74] and treat the obtained output sentences as pseudo-references. Then, we train an autoregressive GPT2 as the text generator [45] by word-level cross-entropy (CE) supervised learning, which enables our model to learn quickly. In the second stage, the GPT2 performs

beam search and the output is 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 4.1 provides an overview of the two stages of search and learning in TGLS.

4.2.1 Simulated Annealing Search

The search-based text generation [72, 74] relies on a heuristic-based objective function $s(\mathrm{y}|\mathrm{x})$ 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 $s_{\text{lm}}(\mathrm{x})$, semantic compliance $s_{\text{semantic}}(\mathrm{x}, \mathrm{y})$, and other task-specific scorers $s_{\text{task}}(\mathrm{y}, \cdot)$. These individual scorers are combined by the product of experts [139]:

$$s(\mathrm{y}|\mathrm{x}) = s_{\text{lm}}(\mathrm{y}) \cdot s_{\text{semantic}}(\mathrm{x}, \mathrm{y}) \cdot s_{\text{task}}(\mathrm{y}, \cdot). \tag{4.1}$$

We adopt simulated annealing (SA) [74, 140], which performs local stochastic search to maximize the objective. Concretely, SA starts from an initial candidate output sentence $\mathrm{y}^{(0)}$, 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 $\mathrm{y}^{(t)}$, namely, word insertion, deletion, and replacement. The proposal y' is accepted with probability $p(\text{accept}|\mathrm{y}', \mathrm{y}^{(t)}, \mathrm{x}, T) = \min\{1, \exp(\frac{s(\mathrm{y}'|\mathrm{x}) - s(\mathrm{y}^{(t)}|\mathrm{x})}{T})\}$. Then, $\mathrm{y}^{(t+1)} = \mathrm{y}'$ if y' is accepted, or otherwise, $\mathrm{y}^{(t+1)} = \mathrm{y}^{(t)}$. 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 [74], the objective function and proposal are largely redesigned, detailed below.

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 [72, 74]. In our work, we use the large-scale pretrained GPT2 model [45]. For an output $\mathrm{y} = y_1 \cdots y_n$, the language fluency scorer is the joint likelihood of y, given by $s_{\text{lm}}(\mathrm{y}) = (\prod_{i=1}^{n} p(y_i|y_1, \cdots, y_{i-1}))^{\alpha}$, where α is a hyperparameter

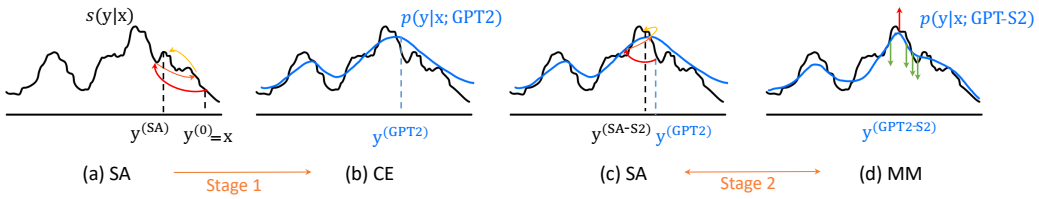

Figure 4.1: Overview of TGLS. (a) First-stage search by simulated annealing (SA). (b) First-stage learning by cross-entropy (CE) loss. (c) Second-stage search by SA. (d) Second-stage learning by max-margin (MM) loss. The horizontal axis represents the sentence space.

balancing s_{lm} with other scorers in (4.1). 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 [74] with a RoBERTa [47]. Fine-tuning details are presented in Section 4.3.3. Compared with autoregressive GPT2 used for fluency evaluation, RoBERTa is pretrained by masked language modeling, and is better at feature representation. Let $\mathrm{x} = (x_1, \cdots, x_m)$ be a sentence. RoBERTa computes a contextualized representation of a word in the sentence as $\text{RoBERTa}(x_i, \mathrm{x})$.

A word-level semantic scorer evaluates how much keyword information (detected by Rake [141]) is preserved, given by the

least matched keyword of x:

$$s_{\text{word}}(\mathrm{y}, \mathrm{x}) = \min_{k \in \text{keyword}(\mathrm{x})} \max_{y_i \in \mathrm{y}} \text{RoBERTa}(k, \mathrm{x})^\top \text{RoBERTa}(y_i, \mathrm{y}). \tag{4.2}$$

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

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

$$s_{\text{semantic}}(\mathrm{y}, \mathrm{x}) = \mathrm{s}_{\text{word}}(\mathrm{y}, \mathrm{x})^\beta \cdot \mathrm{s}_{\text{sent}}(\mathrm{y}, \mathrm{x})^\gamma, \tag{4.3}$$

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 [74] uses the BLEU score to penalize the n-gram overlapping between the output and input: $s_{\text{paraphrase}}(\mathrm{y}, \mathrm{x}) = (1 - \text{BLEU}(\mathrm{y}, \mathrm{x}))^\delta$, 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 [21], which is an example of text style transfer. We follow the setting of most text style-transfer work [17], 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 $s_{\text{formality}}(\text{y}) = p(\text{formal} \,|\, \text{RoBERTa}([\text{BOS}], \text{y}))^{\delta}$, where δ is the weighting hyparaparameter for this task.

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

For `Replace`, the model suggests a candidate word at x_i based on the posterior distribution induced by $s(\text{y}|\text{x})$. For efficiency concerns, previous work [72, 74] 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 also have rule-based local edits (e.g., "*we are*" substituting "*we're*") which are retrieved from PPDB [142]. Previous sequence-to-sequence approaches on this task adopt manually designed rules as a preprocessing step [21] or as additional input concatenated with the raw sentence [23]. Our unsupervised TGLS, on the other hand, can easily make use of the off-the-shelf resources, and can easily filter out the noise by rejecting bad candidates.

In short, the SA search component in our TGLS mainly follows [74], but we re-design the scoring functions and the proposals. The main focus is to couple search and learning, especially the methods of training a machine learning model that learns from the search results, as follows.

4.2.2 Word-Level Cross-Entropy (CE) Learning

As mentioned in Section 4.1, 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, we concatenate an input x and SA's searched sequence $\mathrm{y}^{(\mathrm{SA})}$ with a special separating token [SEP] in between, and train GPT2 with losses on the y-part. Therefore, the GPT2 would be able to generate an output sequence directly from $p(\mathrm{y}|\mathrm{x})$ in an autoregressive way.

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

$$J_{\mathrm{CE}} = -\sum_{i=1}^{N} \sum_{v \in \mathcal{V}} y_{i,v}^{(\mathrm{SA})} \log p_{i,v}^{(\mathrm{GPT2})}, \tag{4.4}$$

where $y_{i,v}^{(\mathrm{SA})}$ is a binary value, indicating whether the ith word is v or not in the SA's output for this data sample, N is the length of y, and $p_{i,v}^{(\mathrm{GPT2})} = \Pr\left[y_i = v \,|\, \mathrm{y}_{<i}^{(\mathrm{SA})}, \mathrm{x}\right]$, 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 $p_{i,v}^{(\text{GPT2})}$ at step i, assuming all previous words are correctly predicted as $\text{y}_{<i}^{(\text{SA})}$. 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 (4.4) is equivalent to minimizing $\text{KL}(\widehat{\boldsymbol{y}}_i^{(\text{SA})} \| \boldsymbol{p}_i^{(\text{GPT2})})$, i.e., the KL-divergence between $\widehat{\boldsymbol{y}}_i^{(\text{SA})}$ and $\boldsymbol{p}_i^{(\text{GPT2})}$, if viewed as distributions over the vocabulary. Due to the asymmetry nature, minimizing the KL-term makes the second slot $\boldsymbol{p}_i^{(\text{GPT2})}$ more wide-spreading than the first slot $\widehat{\boldsymbol{y}}_i^{(\text{SA})}$, illustrated in Figure 4.1(b). 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.

4.2.3 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.,

$\mathrm{y}^{(0)} = \mathrm{x}$), 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 another stage of search and learning alternations. SA starts from an initial candidate being GPT2's output, i.e., $\mathrm{y}^{(0)} = \mathrm{y}^{(\mathrm{GPT2})}$, shown in Figure 4.1(c). 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, in this stage. Such alternation can be performed for multiple epochs for performance bootstrapping.

Concretely, the GPT2 trained with CE learning performs beam search (beam size B) and obtain a set of output sequences $Y^{(\mathrm{GPT2})} = \{\mathrm{y}^{(\mathrm{GPT2},1)}, \cdots, \mathrm{y}^{(\mathrm{GPT2},B)}\}$. A randomly picked (for efficiency purpose) output in $Y^{(\mathrm{GPT2})}$ is taken as initial candidate in SA search, yielding a new sample $\mathrm{y}^{(\text{SA-S2})}$. We consider the set $\widetilde{Y} = Y^{(\mathrm{GPT2})} \cup \{\mathrm{y}^{(\text{SA-S2})}\}$ as the positive and negative samples for MM learning. In fact, the positive sample y^{+} is the best sequence scored by (4.1), i.e., $\mathrm{y}^{+} = \arg\max_{\mathrm{y}\in\widetilde{Y}} s(\mathrm{y}|\mathrm{x})$. In most cases, we have

$\mathrm{y}^+ = \mathrm{y}^{(\text{SA-S2})}$, but this is not necessarily true because SA is not greedy. All other sentences in $\widetilde{Y}$ are collected as negative samples. We use the average of GPT2's pre-softmax logit as the negative energy.[1] In other words, we have $-E(\mathrm{y}) = \frac{1}{N}\sum_{i=1}^{N} z_{i,y_i}$ of a sequence $\mathrm{y} = (y_1, \cdots, y_N)$, where z_{i,y_i} is the logit for the word y_i at the ith step. The max-margin loss for this data sample is

$$J_{\text{MM}} = \sum_{\mathrm{y}^- \in \widetilde{Y}, \mathrm{y}^- \neq \mathrm{y}^+} \max\big\{0, E(\mathrm{y}^+) - E(\mathrm{y}^-) + \Delta\big\}, \qquad (4.5)$$

where Δ (set to 1) is the margin hyperparameter.

In fact, the energy implicitly defines a globally normalized distribution as $p(\mathrm{y}) = \frac{1}{Z}\exp\{-E(\mathrm{y})\}$, 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 (4.1) in its probable region of the output space, illustrated in Figure 4.1(d).

By contrast, word-level CE increases the probability of the target (analogous to the positive sample) step-by-step, while decreasing

[1] *Energy* is what MM learning would like to minimize for positive samples.

Algorithm 1: Training TGLS

Input: A non-parallel corpus X
Output: A fine-tuned GPT2 model
▷ First-stage learning from search
for *an input* $\mathrm{x} \in X$ **do**
  $\mathrm{y}^{(\mathrm{SA})} = \mathrm{SA}(\mathrm{x}, \mathrm{x})$
    ▷ 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* $\mathrm{y}^{(SA)}$ **do**
    Fine-tune GPT2 by cross-entropy loss (4.4) with pseudo-reference $\mathrm{y}^{(\mathrm{SA})}$, conditioned on x
▷ Second-stage learning from search
for *all epochs* **do**
  for *an input* x **do**
    $Y^{(\mathrm{GPT2})} = \mathrm{BeamSearch}(\mathrm{GPT2}(\mathrm{x}))$ ▷ $Y^{(\mathrm{GPT2})}$ is a set of output by beam search
    $\mathrm{y}^{(\mathrm{SA\text{-}S2})} = \mathrm{SA}(\mathrm{x}, \mathrm{y}^{(\mathrm{GPT2})})$ for some $\mathrm{y}^{(\mathrm{GPT2})} \in Y^{(\mathrm{GPT2})}$
      ▷ In the second stage, SA starts with GPT2's output (any output in the beam is fine)
    $\widetilde{Y} = Y^{(\mathrm{GPT2})} \cup \{\mathrm{y}^{(\mathrm{SA\text{-}S2})}\}$
    Fine-tune GPT2 with max-margin loss (4.5) with
      positive sample: $\mathrm{y}^{+} = \arg\max_{\mathrm{y} \in \widetilde{Y}} s(\mathrm{y}|\mathrm{x})$, and
      negative samples: $\widetilde{Y} \backslash \{\mathrm{y}^{+}\}$
Return: Resulting GPT2 (denoted by GPT2-S2 after two stages of search and learning)

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

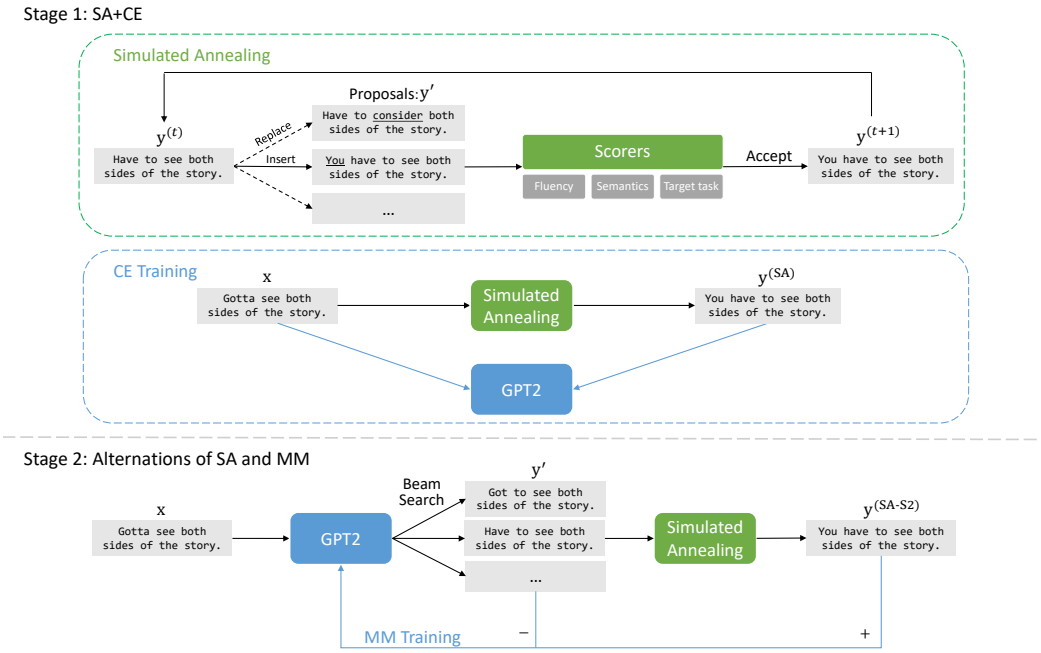

Figure 4.2: Two stages of search and learning in TGLS.

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.

Figure 4.2 shows a detailed diagram of TGLS. Algorithm 1 further presents the pseudo-code of SA search [74] for reference.

4.2.4 Discussion: TGLS vs. Reinforcement Learning and Structured Prediction

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 [143] or an adversarial discriminator [144]) 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 [144]. 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 grountruth for pretraining. We instead perform SA search and learn from SA's (local) optima step-by-step.

Monte-Carlo Tree Search (MCTS) [145] 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 [146, 147] and text generation [148, 149]. L2S allows the model to explore/search in the space, collects the score (cost) for possible actions, and optimizes the model. Usually, L2S assumes that an expert demonstration (groundtruth sequence and/or dynamic oracle) is available as a reference policy. 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 [146, 148]. By contrast, TGLS does not have groundtruth, but uses a strong search algorithm to find higher-scored sentences, which serve as positive samples.

Our approach is also related to learning an inference network for energy-based structured prediction [150, 151]. They perform adversarial learning on the energy model (analogous to a discrimi-

nator) and the inference network (analogous to a generator), with the access of groundtruth target. We instead face an unsupervised setting, where we define the heuristic scorer for discrete search; our conditional generator further learns from the search results.

4.3 Experimental Setting

4.3.1 Datasets and Metrics

Paraphrase Generation. Paraphrase generation is to rephrase input text with different expressions, while keeping the semantics. Following previous work [152, 153], we conducted experiments on the Quora benchmark dataset.[2] We followed the unsupervised setting in [74] 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 [109] penalizes BLEU by

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

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 a given informal text into the formal style. We experimented with the Grammarly's Yahoo Answers Formality Corpus (GYAFC) [21] 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 [17], 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 [154]. 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 separately trained classifier predicts the formality accuracy. Particularly, the formality evaluator achieves an accuracy of 94%, being a good automatic evaluation measure.[3] The BLEU score is also computed against the reference to evaluate

[3]We 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.

Algorithm 2: SA for Text Generation [74]

Input: An input sentence x,
An initial candidate output $\mathrm{y}^{(0)}$
Output: An output sentence y
for $t = 1, \cdots,$ MaxStep **do**
Set temperature $T = \max\{T_{\text{init}} - C \cdot t, 0\}$ ▷ T_{init}, C: annealing hyperparameters
Randomly pick an edit operator Op $\in$ { `Delete`, `Insert`, `Replace`}
Randomly pick an edit position k
Propose a new candidate $\mathrm{y}' = \mathrm{Op}(\mathrm{y}^{(t-1)}, k)$
Compute acceptance rate
$p(\text{accept}|\mathrm{y}', \mathrm{y}^{(t-1)}, \mathrm{x}, T) = \min\{1, \exp(\frac{s(\mathrm{y}'|\mathrm{x}) - s(\mathrm{y}^{(t-1)}|\mathrm{x})}{T})\}$
$$\mathrm{y}^{(t)} = \begin{cases} \mathrm{y}', & \text{with probability } p(\text{accept}|\mathrm{y}', \mathrm{y}^{(t-1)}, \mathrm{x}, T) \\ \mathrm{y}^{(t-1)}, & \text{with probability } 1 - p(\text{accept}|\mathrm{y}', \mathrm{y}^{(t-1)}, \mathrm{x}, T) \end{cases}$$
Return: $\mathrm{y}^{(\text{SA})} = \text{argmax}_t\, s(\mathrm{y}^{(t)}|\mathrm{x})$

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.

4.3.2 Baseline Methods

We present more details on the competing methods in Tables 4.2 and 4.3. All metrics are computed based on word-level tokenization (i.e., no BPE segmentation is used).

We adopt the following baseline methods in paraphrase generation:

(i) **RL-NN.** Qian et al. [65] propose to learn a reward function

by neural networks and perform REINFORCE training. The results in Table 4.2 are from the original paper.

(ii) **DAGGER.**† Du et al. [67] apply imitation learning to paraphrase generation. It achieves the state-of-the-art performance on the Quora dataset. We re-ran their implementation based on our data split.

(iii) **GPT2.**† We train another GPT2 with the same hyperparameters as our TGLS, but in a supervised setting for a controlled comparison.

(iv) **Round-Trip MT (Transformer)**. Following [69], we utilize a well-trained bi-directional neural machine translation (NMT) system (Zh→En and En→Zh) with a Transformer model [31]. 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 round-trip translated sentence (En→Zh→En) as the paraphrase.

(v) **Round-Trip MT (GPT2)**. Similarly, we adopt another GPT2-based multilingual (En, Zh, Es, Ru) NMT system in [70]. Suggested by Guo et al. [70], we take the Zh as the

pivot language.

(vi) **VAE.** The variational autoencoder [155] 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 [74].

(vii) **CGMH.** Miao et al. [72] propose a word-space Metropolis–Hastings approach to paraphrase generation. Results are also quoted from the implementation of [74].

(viii) **UPSA.** Liu et al. [74] extend CGMH by decreasing the temperature and this becomes simulated annealing. Results are quoted from the original paper.

(ix) **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.

We consider the following approaches as our baseline in text style transfer:

(i) **LSTM-attn.** Rao et al. [21] trained a Bi-LSTM encoder-decoder model with attention to their parallel formality corpus.

(ii) **BackTrans.** Prabhumoye et al [156] utilize back-translation to get style-independent content representations and feed them to a style-dependent decoder to control the style of output.

(iii) **StyleEmb.** Fu et al. [82] propose two variants for style

transfer. In this variant, they accomplish style transfer by a learned style embedding.

(iv) **MultiDec.** The other variant of [82] use multiple decoders for style-specific generation.

(v) **CrossAlign.** Shen et al. [81] also use style embedding, but they apply adversarial training based on style-transferred hidden states to cross-align content.

(vi) **DelRetrGen.** Li et al. [20] propose a heuristic 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.

(vii) **Template.** This is a simpler variant in [20]. Then the detected style-specific words of input sentences are replaced by stylized words of the target domain within its retrieved counterpart.

(viii) **UnsupMT.** Zhang et al. [157] 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

Table 4.1: Human evaluation on the Quora dataset.

Method	Grammar, Fluency	Coherency, Consistency	Agreement
UPSA [74]	4.05	3.28	35.0%
SA w/ PLM	4.79	4.48	70.0%
Our TGLS	**4.85**	**4.66**	78.8%

training.

(ix) **DualRL.** Luo et al. [158] use a dual reinforcement learning strategy to learn bi-directional style transfer without explicitly separating the style and content.

The results in Table 4.3 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 [21], and the rest outputs are published by [158].

4.3.3 Implementation Details

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 search and learning for another 6 epochs.

Additionally, we fine-tune a RoBERTa as the backbone model, which is used for scoring semantic compliance and the proposal of replacing words in simulated annealing (Section 4.2.1). In particular, we consider two fine-tuning objectives, as follows.

Masked language modeling. This fine-tuning objective is based on domain-specific unlabeled corpora, and its goal is to adapt RoBERTa and make it 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 [47], which randomly masks out a few words in a sentence, and the fine-tuning 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 the text formalization experiment, where RoBERTa is also used for training a formality classifier. The objective is cross-entropy loss between $p(\text{formal} \,|\, \text{RoBERTa}(\text{BOS}, \text{x}))$ and the groundtruth formality label (formal or informal), where $\text{RoBERTa}(\text{BOS}, \text{x})$ 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.

Conditional generator. We fine-tune the GPT2 model with

Table 4.2: Automatic evaluation results on paraphrasing. † indicates that the results are directly comparable to TGLS on the same data split.

Methods	iBLEU	BLEU
Supervised		
RL-NN [65]	14.83	20.98
DAGGER† [67]	18.88	28.42
GPT2† [45]	19.19	26.92
Distant supervised		
Round-Trip MT (GPT2)† [70]	11.24	16.33
Round-Trip MT (Transformer)† [69]	14.36	20.85
Unsupervised		
VAE [155]	8.16	13.96
CGMH [72]	9.94	15.73
UPSA [74]	12.02	18.18
SA w/ PLM (Ours)†	14.52	21.08
TGLS (Ours)†	**17.48**	**25.00**

task-specific data. For paraphrasing, we use all training sentences, whereas for style transfer, we use the training set of the formal style only.

For fine-tuning hyperparameters, 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.

Table 4.3: Automatic evaluation results on formality transfer. ↓The smaller, the better. † indicates that the results are directly comparable to TGLS on the same data split.

Methods†	PPL↓	BLEU	Formality	H-mean	G-mean
Supervised					
LSTM-attn [21]	**23.42**	**69.36**	**87.39**	**77.34**	**77.85**
Unsupervised					
BackTrans [156]	183.7	1.23	31.18	2.37	6.13
StyleEmb [82]	114.6	8.14	12.31	9.80	10.01
MultiDec [82]	187.2	13.29	8.18	10.13	10.42
CrossAlign [81]	44.78	3.34	67.34	6.36	14.99
DelRetrGen [20]	88.52	24.95	56.96	34.70	37.69
Template [20]	197.5	43.45	37.09	40.02	40.14
UnsupMT [157]	55.16	39.28	66.29	49.33	51.02
DualRL [158]	66.96	54.18	58.26	56.15	56.18
TGLS (Ours)	**30.26**	**60.25**	**75.15**	**66.88**	**67.29**

4.4 Results and Analysis

4.4.1 Overall Performance

Table 4.2 presents the results of automatic evaluation for paraphrase generation. Among the unsupervised approaches, the simulated annealing model UPSA [74] achieves the previous state-of-the-art performance, outperforming both variational sampling [155] and discrete-space Metropolis–Hastings sampling [72]. We propose to use large-scale pretrained language models for fluency and evaluation (denoted by SA w/ PLM), and improve iBLEU by 2.5 points from UPSA. Our TGLS framework of search and learning fur-

ther 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 models [65, 67]. 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 4.3 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 [20], unsupervised machine translation approaches [157], and dual reinforcement

learning [158].

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.

Human Evaluation. We also 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 & fluency) in a blind manner. Each example was reviewed by two judges and average scores are reported in Table 4.1. 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 pre-

Table 4.4: 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?

trained language models for fluency and semantic scorers—indeed improves human satisfaction in terms of Grammar/Fluency and Coherence/Consistency. Our TGLS model further improves the performance in both aspects. The results are also consistent with the automatic metrics in Section 4.4.1.

4.4.2 Ablation Study

As TGLS involves two stages of search and learning, we conduct an ablation study, shown in Table 4.5. 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 [159]. 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 4.5 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

Table 4.5: 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

output region.

4.4.3 Case Study

We show in Table 4.4 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.

4.4.4 Efficiency Analysis

We report inference time in Table 4.5. The experiments were conducted on a cluster with Nvidia Telsa 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, SA+CE+SA+CE, and SA+CE+SA+MM 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 others 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.

The training efficiency of TGLS is roughly twice as much as SA plus GPT2 fine-tuning. We do not have quantitative comparison,

because training efficiency highly depends on hyperparameters and early stop strategies. While our training is more complex than SA or GPT2, we do not view it as a disadvantage. First, training is usually done offline; when trained, our model is very efficient for deployment compared with SA. Second, it is understandable that we sacrifice some training efficiency compared with supervised models, since we do not have parallel data. In fact, our approach should be more efficient (and labor-saving) than data collection plus human annotation in the supervised setting.

4.5 Summary

This work proposes a novel learning-from-search framework TGLS for unsupervised text generation. We show that the simulated annealing search can provide high-quality examples for training a conditional text generator. Further, the 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.

□ **End of chapter.**

Chapter 5

Text Revision by On-the-Fly Representation Optimization

In this chapter, we endeavour to explore an an efficient way to resolve unsupervised controllable text generation. Specifically, we focus on revision tasks, where we aim to revise the input text in place, rather than generating it token by token from scratch" would be clearer and more concise. The background review and motivation of this idea in Section 5.1, followed by problem formulation in Section 5.2. The preliminary of base models and the proposed framework are detailed in Section 5.3. And the experimental setup is explained in Section 5.4. The results and analysis are detailed in Section 5.5. To sum up, Section 5.6 reviews the contributions

of this chapter.

5.1 Introduction

Text revision refers to an important series of text generation tasks, including but not limited to text style transfer [81], text simplification [94], counterfactual debiasing [160], grammar error correction [161], sentence fusion [162] and argument reframing [163], which revises an input sentence into another one with the desired attribute (e.g., formality or simplicity). As the most popular solution, sequence-to-sequence (seq2seq) learning achieves state-of-the-art results on many text revision tasks today. However, it becomes less applicable when there is no large-scale annotated parallel data for training.

On the other hand, recent breakthroughs in self-supervised learning have enabled the pre-trained Transformer (TFM) models [36], such as BERT [46], RoBERTa [47] and GPT [44], to learn sufficient distributed representation of natural language, which is universally transferable to a wide range of downstream tasks even without labeled data [106, 164, 165]. In this chapter, we ask the question, can we borrow the power of a pre-trained Transformer

for text revision without any parallel data?

There exist some efforts on developing unsupervised text generation methods with only non-parallel data, such as using reinforcement learning (RL) [166] and variational auto-encoders [17]. However, these methods suffer from issues of unstable [167] and computationally expensive training. It is even more challenging to apply them with large pre-trained models. For instance, to fine-tune a GPT-3 summarization model with RL, it takes thousands of labeler hours for learning a reliable reward function and 320 GPU-days to train the policy and value nets [34].

In this work, we propose OREO, a method of On-the-fly REpresentation Optimization for text revision. Instead of generating an entire sequence of tokens from scratch, OREO first detects partial text span to be edited, then conducts in-place span revision, which is realized by iterative mask-and-infill editing on the input sentence. As shown in Figure 5.1, at each iteration, a fine-tuned RoBERTa encodes the input sentence into a distributed representation, then optimizes it informed by an attribute head of the same pretrained RoBERTa model. After that, OREO masks a span and infills a new one conditioned on the updated representa-

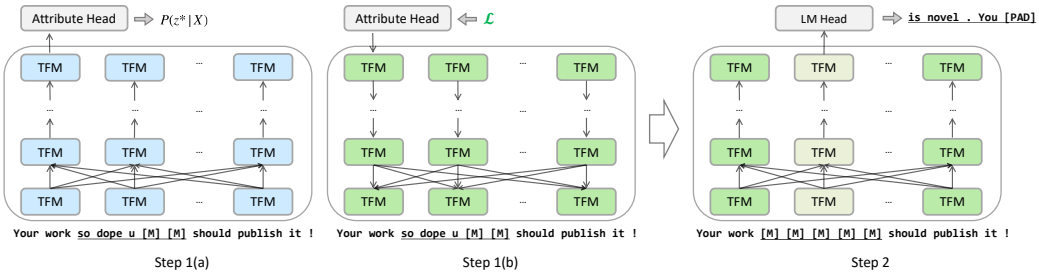

Figure 5.1: A simplified illustration of two-step span revision in OREO. In this example, the input is "*Your work so dope u should publish it!*". The informal textual span "*so dope u*" is selected to revise. To allow for a potentially longer replacement, we append 2 `[LM-MASK]` tokens to the span and use this sequence for a two-step revision. Step 1: Representation Optimization. (a) The fine-tuned RoBERTa model encodes an input sentence to calculate the likelihood of target attribute $P_\theta(z^*|X)$. (b) After calculating and backpropagating the loss between estimated and target attribute values, the hidden states (in green) are optimized on the fly. Step 2: Span replacement. The span to be edited is replaced with `[LM-MASK]` tokens (we use `[M]` for short). We fix the optimized hidden representations in Step 1 (in green) and let RoBERTa's LM head propose an alternative text span autoregressively.

tion. As for the training, our model, OREO fine-tunes RoBERTa with two simple tasks, masked language modeling and attribute classification.

The contribution of this work is three-fold:

1. We propose an efficient mask-and-infill method with on-the-fly optimized representation for text revision. In this work, we tackle two important tasks: text simplification and text formalization. Additionally, this framework can be directly adapted to other text revision tasks.

2. To enable on-the-fly representation optimization, we design

simple fine-tuning methods that balance efficiency and efficacy. The fine-tuning can be finished within 8 GPU-hours at most in our experiments.

3. Our proposed OREO has strong performance on text formalization dataset GYAFC-fr [21], surpassing unsupervised baseline methods, one of which also utilizes RoBERTa; and achieves competitive performance with state-of-the-art supervised methods on text simplification dataset NEWSELA-TURK [98].

5.2 Problem Formulation

Text revision aims to revise an input sentence X with attribute z to another one X^* with the target attribute z^*, while other features are fixed as much as possible. In this work, we address text simplification and text formalization, where the target attributes are simplicity and formality respectively. The training data is a non-parallel corpus with attribute labels.

5.3 Methodology

5.3.1 Preliminary: Pre-trained TFM Models for Natural Language

Self-supervised learning with massive unlabeled text data makes powerful pre-trained Transformers for natural language processing. We adopt the $\text{RoBERTa}_{\text{base}}$ [47] model in this work.

RoBERTa is a stack of L Transformer layers trained with masked language modeling with unlabeled text data. Given a sequence of tokens $[x_1, ..., x_T]$ with length T that is partially masked (e.g. x_t is replaced by a special `[MASK]` token), RoBERTa constructs hidden states H_t^l at l-th layer for a token x_t. On top of the Transformer layers of RoBERTa, there is a language model (LM) head that takes as input the hidden states H_t^L at the final layer corresponding to the masked token, and recovers the masked token x_t by maximizing:

$$P_{W_{\text{LM}}}(x_t|H_t^L) = \text{Softmax}(W_{\text{LM}}^T H_t^L), \tag{5.1}$$

where W_{LM} is the parameter of LM head and $H_{\setminus t}$ is hidden states at positions other than t. H_t has intensive interaction with $H_{\setminus t}$

through self-attention module. Therefore, RoBERTa is able to infill context-aware tokens.

5.3.2 Training: Multi-task Fine-tuning

The hidden states produced by RoBERTa, or in general, pre-trained Transformer models, have been proven to encode a wide range of linguistic features, such as morphology [168], syntax [165], semantics [106] and etc. Motivated by this, we fine-tune the RoBERTa to model the task-specific attributes. Concretely, we adopt two fine-tuning tasks, masked language modeling (MLM) and attribute classification. The former one is to force RoBERTa to infill a span consistent with the semantics and attributes encoded in the hidden states, and the latter one is to help RoBERTa update the hidden states towards a specific attribute.

Masked language modeling

The original MLM objective adopted by RoBERTa does not model the length of tokens to be infilled. Inspired by Malmi et al. [83], we let the model do variant-length span replacement. Specifically, there are three modifications for the MLM objective: 1) We intro-

duce a new special token `[LM-MASK]` for span infilling; 2) Before each iteration of span replacement, we append K additional masks to pad the selected span to a fixed length; 3) RoBERTa can predict `[PAD]`, another new special token, as a placeholder to be removed directly from the output text. As such, a selected span of length N can be replaced by a new one, whose length is between 0 and $N+K$.

We modify the strategy for MLM training data construction accordingly. A continuous span is masked, and we randomly insert `[LM-MASK]` and `[PAD]` tokens in the source and target spans, respectively.

Meanwhile, we still follow the original masking strategy, where tokens are masked independently and replaced by `[MASK]` token, creating another set of MLM training data. We fine-tune RoBERTa and its LM head with two sets of training data jointly.

Attribute classification

In addition, we create a new attribute head, parallel to the LM head, on top of RoBERTa as an attribute classifier. The conventional fine-tuning approach takes as input the outputs of the

final layer at position $t = 0$. In our preliminary experiment, we find this approach sub-optimal. Inspired by the evidence found in [164] that the different layers of pre-trained Transformer capture different categories of features, we concatenate the hidden states of the `[CLS]` token from all layers as the input of attribute head. Specifically, given an input sentence X, RoBERTa with parameters θ predicts the probability distribution over attribute candidates Z as:

$$P_\theta(Z|X) = \text{Softmax}(W_{\text{Att}}^T[H_0^0, H_0^1, ..., H_0^L]) \tag{5.2}$$

where W_{Att} denotes parameters of the attribute head, and $[H_0^0, H_0^1, ..., H_0^L]$ is the concatenation of hidden states from all layers at the position $t = 0$. Then the RoBERTa is tuned to maximize the likelihood of ground-truth attribute labels.

5.3.3 Inference: On-the-fly Representation Optimization

Most of the existing work on unsupervised text generation incorporate task-specific constraints, such as reconstruction objective and discriminator networks [100], on the generation model explicitly. In contrast, we steer the distributed representation of text directly. The hypothesis is that the pre-training and fine-tuning

Algorithm 3: Text revision with OREO

Input: An input sentence $\mathrm{X}^{(0)}$;
Set target attribute z^*, threshold δ, maximum iteration number I;
A fine-tuned RoBERTa with parameters θ, including an attribute head W_{Att} and a LM head W_{LM}

Output: An output sentence X^*

Initialize: $i = 0$, $\zeta^{(0)} = P_\theta(z^*|X^{(0)})$

while $i < I$ and $\zeta^{(i)} < \delta$ **do**

- ▶ Span selection
- Calculate $\zeta^{(i)} = P_\theta(z^*|X^{(i)})$ and $\mathcal{L}$ (5.4)
- Calculate $a^{(i)}$ (5.6) and select $t, N = \arg\max_{t,N} a^{(i)}_{t:t+N}$
- ▶ Representation optimization
- Insert K `[LM-MASK]`s after $X^{(i)}_{t:t+N}$, then we have $X'^{(i)}$ as the input of RoBERTa at the next step
- Calculate $H^{(i)}$, $P_{W_{\text{Att}}}(z^*|H^{(i)})$ and $\mathcal{L}'$ (5.4)
- Update $H^{(i+1)}$ with $\nabla_{H^{(i)}}\mathcal{L}'$ (5.3)
- ▶ Span replacement
- Replace the selected span $X'^{(i)}_{t:t+N}$ with `[LM-MASK]`s
- $X^{(i+1)}_{\setminus t:t+N+K} = X'^{(i)}_{\setminus t:t+N+K}$
  - ▶ The unselected part keep fixed
- Infill a new span $X^{(i+1)}_{t:t+N+K} = \arg\max_{X_{t:t+N+K}} P_{W_{\text{LM}}}(X_{t:t+N+K}|H^{(i+1)}_{\setminus t:t+N+K})$
  - ▶ Approximate by greedy decoding
- Remove the `[PAD]` tokens in the new span, then we have $X^{(i+1)}$

Return: $X^* = X^{(j)}$, where $j = \arg\max_j \zeta^{(j)}$

make RoBERTa an intrinsic multi-task model, which has already learned sufficient features for text revision: the hidden states can be used to recognize the attribute, and meanwhile inform the LM head to select tokens consistent to a certain attribute and context. All we need further is to keep other attributes, especially the semantics, fixed as much as possible during modification.

To this end, OREO conducts text revision by iteratively replacing spans on the input sequence. At each iteration, a span is selected for editing; then the revision is done in two steps. At the first step, RoBERTa encodes the input sentence into hidden states, conditioned on which the attribute head measures the probability of target attributes. Then RoBERTa adjusts the hidden states towards increasing the target attribute probability. At the second step, the selected span is masked out, after which RoBERTa uses the LM head to fill in the blank, conditioned on updated hidden states. These two steps repeatedly iterate until a maximum iteration number I is reached, or the attribute value exceeds a predefined threshold δ. The complete revision procedure of OREO is formalized in Algorithm 3.

In the following sections, we detail two steps of text revision in OREO respectively. An illustration is provided in Figure 5.1. Then we introduce our method of span selection.

Step 1: Representation optimization

Given an input sentence $X^{(i)}$ at the i-th iteration, RoBERTa parameterized by θ transforms it to a sequence of hidden states $H^{(i)}$,

conditioned on which the attribute head estimates the probability of target attribute $P_{W_{\text{Att}}}(z^*|H^{(i)})$. However, blindly finding a H^* that optimizes $P_{W_{\text{Att}}}(z^*|H^*)$ can corrupt or even eliminate other useful features encoded in the original hidden states, and we may not want those features to be greatly influenced. Thus, for each revision, we find a small local perturbation on $H^{(i)}$ that maximally increases the likelihood of target attribute. As such, the update rule of hidden states is:

$$H^{(i+1)} = H^{(i)} - \lambda \frac{\nabla_{H^{(i)}} \mathcal{L}}{\|\nabla_{H^{(i)}} \mathcal{L}\|_2}, \tag{5.3}$$

where λ is a hyper-parameter that controls the norm of perturbation, and

$$\mathcal{L} = -\log P_{W_{\text{Att}}}(z^*|H^{(i)}). \tag{5.4}$$

The perturbation, also known as the normalized gradient of $\mathcal{L}$ with respect to hidden states, can be calculated with standard backpropagation techniques. The parameters of RoBERTa is frozen during this gradient computation. Therefore, the representation is optimized on-the-fly.

Even though we apply a small perturbation, there are still risks that other coupled attributes change accordingly. We address this issue by only replacing one span at each iteration, and encoding

the complete sentence into hidden states before masking a span. This issue can be further eliminated by other advanced techniques, such as representation disentanglement [169] and neural adapter modules [170]. We leave the exploration of more advanced solutions for future work.

Step 2: Span replacement

Once the hidden states are updated, OREO conducts span replacement. The selected span $X^{(i)}_{t:t+N}$ of length N is replaced by `[LM-MASK]` tokens. And hence the span to be infilled is $X^{(i)}_{t:t+N+K}$ (we append K `[LM-MASK]` tokens before updating hidden states). RoBERTa takes as input the masked sequence, and predicts a new span autoregressively with the previously updated hidden states:

$$P_{W_{\text{LM}}}(X^{(i+1)}_{t:t+N+K}|H^{(i+1)}_{\setminus t:t+N+K}) = \prod_{n=1}^{N+K} P_{W_{\text{LM}}}(x^{(i+1)}_{t+n}|H^{(i+1)}_{\setminus t:t+N+K}, X^{(i+1)}_{t:t+n}), \tag{5.5}$$

where $x^{(i+1)}_{t+n}$ is the predicted token at step n, $H^{(i+1)}_{\setminus t:t+N+K}$ is the optimized hidden states of unselected text. Informed by the updated hidden states, the revised span is expected to meet target attribute and meanwhile maintain other information, e.g. semantics, of the original span.

Span selection strategy

The span selection in OREO is done before the text revision at each iteration. It is motivated by three reasons: 1) The selection strategy can be agnostic to the text revision algorithm, increasing the flexibility of OREO; 2) It allows us to insert `[LM-MASK]` tokens in the selected span in advance, so that RoBERTa can infill a longer span. 3) It enables human-in-the-loop generation, where the user can indicate which part should be revised.

In this work, we use the magnitude of the $\nabla_{H^{(i)}}\mathcal{L}$, where $\mathcal{L}$ is calculated with (5.4), as a measurement of disagreement for span selection. Specifically, at iteration i, we calculate $a_t^{(i)}$ for each token with respect to the attribute head as:

$$a_t^{(i)} = \|\nabla_{H_t^{0(i)}}\mathcal{L}\|_2, \tag{5.6}$$

where H^0 is the hidden states at the word embedding layer. Intuitively, a token whose modification can maximally increase the target attribute value should be revised.

Then we calculate an N-gram ($n \leq 4$) score as:

$$a_{t:t+N}^{(i)} = \frac{\sum_{n=1}^{N} a_{t+n}^{(i)}}{N + c}, \tag{5.7}$$

where we add a smoothing constant c, otherwise only one token is chosen. In practice, we set c as 1. To further prevent serious corruption of the original sentence, we remove named entities from the selected span. As mentioned above, we finally append K `[LM-MASK]` tokens to the selected span for the two-step span replacement.

5.4 Experimental Setting

We experiment with OREO in two real-world text revision tasks, text simplification and text formalization.

5.4.1 Datasets and Metrics

Text Simplification. Text simplification is to revise the complex text into simpler language with easy grammar and word choice while keeping the meaning unchanged [171]. Based on the widely used corpora Newsela [97], Jiang et al. [172] constructs a reliable corpus consisting of 666K complex-simple sentence pairs[1]. As our model does not rely on the complex-simple alignments, we remove the duplicated sentences. The final dataset consists of 269K

[1]Dataset available at https://github.com/chaojiang06/wiki-auto. Newsela dataset can be requested from https://newsela.com/data/

train, 28K development and 29K test sentences. As discussed in [98, 172, 173], previous supervised methods tend to behave conservatively by simply deleting words and lack the ability to conduct effective phrasal simplification, we follow [98] and adopt NEWSELA-TURK for evaluation, a test set with high-quality human-written references emphasizing lexical and phrasal simplification for each complex sentence. Although it is challenging for OREO to conduct structural simplification, there is an off-the-shelf resource [174] focused on sentence splitting and deletion that we can utilize as a pre-processing of complex sentences. To keep this work focused, we leave structural transformation for future work.

We report SARI [94], Flesch-Kincaid grade level (FKGL) readability [175] and average sentence length (SLen) as evaluation metrics. SARI calculates the average of F1/precision of n-grams added, kept and deleted between system output and reference sentences ($n \in \{1, 2, 3, 4\}$). We report the F1 score of each edit operation. FKGL measures the readability of sentences. We do not report BLEU because it does not correlate well with human judgement [94].

Text Formalization. The second task we experiment with is text

Table 5.1: Automatic evaluation results on NEWSELA-TURK. ↓The smaller, the better.

Methods	SARI	Add	Keep	Delete	FKGL↓	SLen
Supervised						
Complex (Input)	22.3	0.0	67.0	0.0	12.8	23.2
TFM_{BERT}	36.0	3.3	54.9	49.8	**8.9**	16.1
EditNTS	37.4	1.6	61.0	49.6	9.5	16.9
Hybird-NG	38.2	2.8	57.0	54.8	10.7	21.6
CtrlSimp	41.0	**3.4**	63.1	56.6	11.5	22.2
Unsupervised						
UNTS	39.9	1.5	60.5	57.7	11.2	22.0
OREO (ours)	**45.2**	2.3	**69.4**	**64.0**	11.4	**23.5**

Table 5.2: Automatic evaluation results on text formalization.

Methods†	BLEU	Formality	H-mean	G-mean
Reference	100.0	95.20	97.49	97.52
CrossAlign	4.77	75.9	8.98	19.03
StyleEmbded	8.71	28.3	13.32	15.70
MultiDec	14.04	21.32	16.93	17.30
UnsupMT	37.36	76.88	50.28	53.59
MASKER	47.73	58.86	52.71	53.00
OREO (ours)	**57.63**	**80.71**	**67.24**	**68.20**

formalization. Since the informal sentence is much noisier than the pre-training data of RoBERTa, this task can test the robustness of our OREO. To compare with previous work, we experimented with the domain of Family & Relationships in Grammarly's Yahoo Answers Formality Corpus (GYAFC-fr) [21]. There are 100K, 5K and 2.5K informal-formal[2] pairs in GYAFC. Again, we only

[2]The informal text in GYAFC is collected from casual chats in web forums. It includes

use non-parallel sentences and their associated formality labels to fine-tune RoBERTa. Considering the gap between informal text and pre-training corpus, we augment the training data with 880K automatically extracted sentences from the same domain by Xu et al. [154].

The evaluation of formalization involves multiple aspects. Following previous literature [158, 176], we report BLEU [177] as the measurement of content preservation and fluency. The formality attribute is evaluated by a separately trained RoBERTa classifier which obtains accuracy at 94% on the validation set. To obtain an overall performance of the system, we calculate the harmonic mean (H-mean) and geometric mean (G-mean) of BLEU and formality accuracy and consider them as the main metric for this task.

5.4.2 Baseline Methods

For text simplification, we compare our OREO to both supervised and unsupervised approaches. For unsupervised baselines, we adopt UNTS [100], which is based on adversarial training and variational auto-encoder. We also compare our model with the

few offensive statements, such as slang, vulgarity, harassment, etc. These statements may cause discomfort or upset to the user of the dataset.

following state-of-the-art supervised methods:

(i) TFM_{BERT} [178], a Transformer whose encoder is initialized with the BERT model.

(ii) EditNTS [95], which models edit operations explicitly with sequence-to-sequence learning.

(iii) Hybrid-NG [96], a hybrid system including a probabilistic model for splitting and deletion, and a monolingual machine translation model for phrase replacement and reordering.

(iv) CtrlSimp [98], the current state-of-the-art method composed of structural simplification module and lexical/phrasal simplification model.

We also report the performance of the strategy that blindly copies the original complex sentence.

For text formalization, we compare OREO with the following widely adopted unsupervised baseline methods:

(i) CrossAlign [81] disentangles the style of text and contents via shared latent space for style revision.

(ii) StyleEmbedded [82] uses an adversarial network to obtain

the style-irrelevant content representation in the encoder and merges it with a style embedding to control the style.

(iii) MultiDec [82] adopts the same method as [82] to remove the style information from input text, but use decoders with different target styles to insert style information in the decoding stage.

(iv) UnsupMT [157] adopts machine translation methods to deliver pseudo training pairs for sequence-to-sequence transduction.

(v) MASKER [83] employs a BERT which masks the span according to the disagreement of language models conditioned on different attributes and fills in a new span for the target attribute.

As a recently proposed unsupervised method for text style transfer, MASKER is closest to OREO. For a fair comparison, we use RoBERTa as their base model. In our preliminary experiment, we find that RoBERTa leads to better performance on text formalization.

5.4.3 Implementation Details

We implement RoBERTa based on Huggingface transformers [179]. For all experiments, we fine-tune the RoBERTa $_{base}$ [47] with a task-specific corpus.

In the fine-tuning stage, for the standard MLM objective, we replace 15% tokens as `[MASK]`. For the padded variant of MLM, we replace one text span with 3 `[LM-MASK]`s for each training instance. If the length of selected span is less than 3, we append `[PAD]` tokens to it as the target of padded MLM. For example, we mask the first two words in sentence "Good luck to you!" as "`[LM-MASK][LM-MASK][LM-MASK]` to you!", and the target is "Good luck `[PAD]` ". We primarily adopted the default hyper-parameters with a fixed learning rate of 5e-5. The numbers of fine-tuning epochs are 6 and 2 for text simplification and formalization, respectively. It takes 8-GPU hours to fine-tune RoBERTa on one Tesla V100 for both tasks.

In the inference stage, the maximum iteration I was set to 4 for efficiency purpose, although the final performance can increase slightly with more iterations. λ was selected from $\{0.8, 1.2, 1.6, 2.0\}$ and set to 1.6. These parameters are validated

only on the text formalization. We do not perform further tuning on text simplification. The attribute threshold δ is task-dependent. It was selected from from $\{0.1, 0.2, \ldots, 0.5\}$ and set to 0.5 for text simplification and 0.3 for text formalization. $K = 1$ for both tasks.

5.5 Results and Analysis

5.5.1 Overall Performance

Text simplification. Table 5.1 presents the automatic evaluation results for text simplification on NEWSELA-TURK. As for the main metric of text simplification, our method achieves the highest SARI score, surpassing the supervised and unsupervised baseline by a large margin. According to [98], `Add` is an important metric to indicate the model's capability in paraphrasing. OREO gains a higher `Add` score than the supervised edit-based method, EditNTS. Although UNTS is on a par with OREO in FKGL scores, its `Add` score is 0.8 points lower than OREO, indicating that our model has a better trade-off between simplicity and meaning preservation as well as fluency. Our method's high score in `Keep` and `Delete` operations demonstrates that gradient-guided span selection can

detect the complex span accurately.

Text formalization. Table 5.2 shows the evaluation results for text formalization. Our approach outperforms all of the unsupervised baseline models in both content preservation and accuracy of style transfer. Notably, the significant margin of OREO and MASKER demonstrates the necessity of hidden states optimization. Although both methods directly conduct span replacement, OREO additionally performs on-the-fly update on hidden representations of its context, which is steered by an attribute head. This leads to a large improvement in formality. Additionally, MASKER proposes phrasal replacement based on an incomplete input, without accessing the semantics of the original span. This leads to semantic loss. While our span infilling is conditioned on the representations encoded the semantics of the original input, OREO has a large improvement on BLEU score.

Human evaluation. To verify the improvement of OREO, we conduct human evaluation on text formalization in Table 5.3. We randomly sample 80 examples from each model's output and human-written reference. Due to the budget limits, we only compare to the baseline that is closest to our work. We invited six

Table 5.3: Human evaluation on text formalization

	Formality	Coherency	Fluency
MASKER	2.74	2.94	3.31
OREO	3.42	3.33	3.41
Human	**3.69**	**3.67**	**3.78**

annotators with advanced linguistic backgrounds to evaluate formality, semantic coherence and language fluency of each sentence in a blind manner. Formality indicates to how much degree the output satisfies the formal attribute. Semantic coherence means whether the output preserves the original semantics of input text. And language fluency measures the grammatical correctness of the output text. Each annotator is asked to provide scores from 1 to 4 for all three criteria. Each sentence is rated by two annotators [3] and we report the averaged ratings. In Table 5.3, OREO is significantly better than MASKER in terms of formality and coherency (p-value < 0.01), which is consistent with automatic evaluation results. However, there is still improvement space for OREO when compared to human reference. Two edit-based methods have the same score of language fluency, mostly because both of them recruit RoBERTa as the base model to propose new span.

[3]The annotators' ratings are positively correlated with p-value < 0.1 across models and metrics.

Table 5.4: Model ablation study on text formalization.

	BLEU	Formality	H-mean	G-mean
Full	**57.63**	**80.71**	**67.24**	**68.20**
(1) Infill w/o $H^{(i)}$	55.50	69.67	61.78	62.18
(2) Update $H^{(i)}$ w/ noise	56.55	69.14	62.21	62.53
(3) Fix $H^{(i)}$	56.47	67.94	61.68	61.94
(4) Random span selection	45.30	55.03	49.69	49.93

5.5.2 Ablation Study

We evaluate different variants of OREO in Table 5.4. To verify the necessity of infilling conditioned on updated hidden states and the gradient information for the update, we compare to variants as 1) without fixing any hidden state when infilling span; 2) updating the hidden states with Gaussian noise; 3) without updating the hidden states. To evaluate the effect of our span selection strategy, we also try (4) randomly selecting span.

With fixed or incorrectly updated hidden states, the formality of revised text drops sharply. It indicates that optimizing hidden states efficiently is crucial to infilling a span that satisfies the target attribute.

When the hidden states are removed, there is a significant drop in terms of the BLEU score due to the loss of semantic information. Both BLEU score and formality drop drastically when the span

is replaced randomly. It indicates that our gradient-guided span selection is helpful in detecting spans that are opposite to the target attribute.

5.5.3 Case Study

Table 5.5 exhibits the examples generated by baseline methods and OREO in both tasks. Compared to other baseline methods, our OREO is able to produce accurate and fluent revision. More surprisingly, it can even conduct knowledgeable revision. For instance, "*a think tank*" is simplified as "*a group that studies people*". OREO also has decent performance encountering noisy text. In Example 3, MASKER fails to correct the abbreviation and typos, while OREO correctly revises "*u*" to "*you*", and "*kno*" to "*know*".

However, we also notice that OREO sometimes fails to hold semantics. For instance, it revises "*critics*" to "*supporters*" in Example 2. This is a common problem that language models are not sensitive to negation. More efforts could be made in future work.

Table 5.5: Examples of outputs from baseline methods and OREO on text simplification and text formalization. Both successful and erroneous cases are reported.

#	**Complex Input**	**UNTS**	**OREO**
1	still, recent trends suggest seattle is doing a better job of holding onto those kids, according to sightline institute, a think tank based in seattle.	still, recent trend suggest seattle is doing a better job of holding *guns* of those kids, according to *unc*, a think tank in seattle.	still, recent studies suggest seattle is doing a better job of holding onto those kids, according to sightline institute, a *group that studies people* in seattle.
2	critics of the program say the eisenhower deportation program's conditions were anything but humane.	critics of the program say the *nsa operation* program's *conditions's* conditions were anything.	some *supporters* of the program say the eisenhower school program's *rules* were anything but for children.
#	**Informal Input**	**MASKER**	**OREO**
3	tell him, and it wouldn't seem psycho cuz u have kno each other for a long time	It wouldn't seem psycho cuz u have kno each other for a long time	Tell him, and it *will not* even seem *awkward you two* have *known* each other for a long time
4	Intellect - a chick with brains is just sexy!	Intellect - is just sexy!	*I think a woman endowed* with brains is just sexy!

Then we explore human-in-the-loop generation, where a user selects a phrase to be replaced; based on which OREO conducts the revision. We find that this interactive generation can help OREO conduct better revision. The examples are demonstrated in Table 5.6. When a user specifies an accurate span, OREO can correctly revise the sentence to a former tone. For example, OREO failed to automatically detect the unmatched parentheses in Example 5, but it correctly edits the random punctuation after the user points it out. Additionally, we also select some imperfect outputs from OREO. Based on its own system outputs, we can obtain better rewrites by explicitly providing spans to be edited.

5.5.4 Inference Efficiency

An obvious concern of OREO is the inference efficiency, given that it updates the hidden states in a large Transformer on the fly and conducts revision in multiple iterations. Therefore, we report the inference speed here. For text formalization, it takes an average of 0.12 second to revise a sentence in one iteration in OREO and 4.18 seconds in MASKER. We argue that this is acceptable given training in OREO is simple and time-saving. Moreover,

Table 5.6: Examples of human-in-the-loop. Input sentences are edited in multiple iterations. The underlined texts are selected span-to-edit. Orange indicates proposed phrasal replacement.

informal: I'm just looking for the girl who wants that time with me **OREO**: I am not just looking for the girl who wants that time with me **1**$_{st}$ **Edit**: I am still looking for the girl who wants that time with me **2**$_{nd}$ **Edit**: I am still looking for the girl who wants to spend time with me
informal: Then if he still doesn't like you, then move on. **OREO**: Then if he still does not not like you, then move on. **1**$_{st}$ **Edit**: Then if he still does not not like you, then move on. **2**$_{nd}$ **Edit**: Then if he still does not appear to like you, then move on. **3**$_{rd}$ **Edit**: However, if he still does not appear to like you, then move on. **4**$_{th}$ **Edit**: However, if he still does not appear to like you, you should move on.
informal: Maybe your young, I don't know. **OREO**: Maybe your young, I do not really know. **1**$_{st}$ **Edit**: Maybe your young, I do not really know. **2**$_{nd}$ **Edit**: Maybe you are young, I do not really know. **3**$_{rd}$ **Edit**: Maybe you are young, I do not know your age.
informal: No matter what he says, Action speaks louder then words **OREO**: No matter what he says, be patient! **1**$_{st}$ **Edit**: No matter what he says, action speaks louder than verbal words
informal: (even if he is rich, dont sell ur self. **OREO**: (even if he is rich, be your own self. **1**$_{st}$ **Edit**: (even if he is rich, do not sell ur self. **2**$_{nd}$ **Edit**: even if he is rich, do not sell ur self. **3**$_{rd}$ **Edit**: even if he is rich, do not sell yourself to him.
informal: I haven't tried it yet, but I am still trying to get up the nerve. **OREO**: I have not even tried it yet, but I am still trying to get up the nerve. **1**$_{st}$ **Edit**:I have not even tried it yet, but I am still trying to get up the nerve. **2**$_{nd}$ **Edit**: I have not even tried it yet, but I am still trying to get up the courage. **3**$_{rd}$ **Edit**: I have not even tried it yet, but I am still trying to find the strength courage. **4**$_{th}$ **Edit**: I have not even tried it yet, but I am still trying to find the courage.
informal: Then see if shes open for a dinner & a movie. **OREO**: Then see if she will accompany you for a dinner or perhaps a movie. **1**$_{st}$ **Edit**: Inquire her if she will accompany you for a dinner or perhaps a movie.
informal: The same guy you wanna be in a relationship with? **OREO**: The same guy you want to be in a relationship with? **1**$_{st}$ **Edit**: The very same guy you want to be in a relationship with? **2**$_{nd}$ **Edit**: Is this the same guy you want to be in a relationship with? **3**$_{rd}$ **Edit**: Is this the same person that you want to be in a relationship with?
informal: if he really didn't like her and did like you, then he would have already dumped her for you. **OREO**: If he really did not like her and did like you, then he would have already dumped her for you. **1**$_{st}$ **Edit**: if he really did not like her and did like you, then he would have already left her for you.

to further reduce the inference duration, we can employ OREO to construct pseudo-parallel datasets, and learn a conventional

sequence generation model as in [83].

5.6 Summary

In this chapter, we propose a new method for text revision with iterative in-place span replacement. With simple fine-tuning methods, the hidden states of RoBERTa can be optimized towards the target attribute on the fly. Both the automatic evaluation and the human evaluation demonstrate the effectiveness of the proposed method in real-world applications, text simplification and text formalization. In the future, we would like to apply this method to more challenging attributes, e.g. modifying syntax for paraphrasing [169] and question generation [180, 181].

□ **End of chapter.**

Chapter 6

Conclusion and Future Work

This chapter begins with a summary of the contributions of this thesis, providing an overview of the principal findings. Then we present a discussion of potential research directions in the field of controllable text generation.

6.1 Conclusion

The recent advancement of deep learning methodologies [31] has substantially elevated the capabilities of text generation techniques [44, 45]. Consequently, the emergence of controllable text generation presents itself as an innovative and noteworthy challenge within the research community. The ability to manipulate the attributes of generated text is crucial for many NLP applications and

has the potential to impact various industries and domains. In this thesis, we investigate the regulation diverse textual attributes in neural controllable text generation, ranging from semantic control to lexicon manipulation.

In Chapter 3, we focus on the semantic fidelity in neural text generation in a supervised setting. We undertake the study of semantic control within the scope of question generation, which necessitates a high degree of semantic relevance between the input and output text. In our study, by separately extracting and encoding the answer-relevant structured text, we can steer the generation system to focus on the answer-relevant context and produce to the point question.

In Chapter 4 and Chapter 5, we explore the control of multiple textual attributes in an unsupervised setting, which is more challenging. In Chapter 4, we proposed a generalized search-and-learning framework. The empirical study is conducted to investigate the control of lexical diversity in the context of unsupervised paraphrasing and the transduction of formality style in text formalization. Attribute transduction is performed in token-level in the initial search stage. Subsequently, in Chapter 5, we

propose a novel unsupervised approach to CTG via phrasal-level edition. Two major control factors, formality and simplicity, are involved in this study. We conduct experiments in the tasks of text formalization and text simplification. Within our study, the proposed unsupervised solution achieves comparable results in the tasks of paraphrase generation and text simplification, and surpasses the baseline methods in text formalization. Notably, our unsupervised solution significantly reduces the need for labeled data, making it a resource-efficient option.

6.2 Future Work

The innovation of deep neural networks, from RNN-based language model to pretrained large-scale language model, has continuously boosted the advance of text generation. Nonetheless, due to the intrinsic black-box nature of neural models, the investigation of controlling the generation procedure to yield text with desired attributes is still quite challenging yet important. In addition to the contributions presented above, this section explores some intriguing research directions related to NCTG.

6.2.1 Exploring Structured Control Codes

In the the preceding chapters, the exploration of semantic fidelity, formality and simplicity control is conducted in the context of unstructured natural text. Nevertheless, in neural text generation, there is large portion of tasks encountering the structured data as input, i.e., data-to-text generation.

Data-to-text generation takes in structured data to generate text that is the verbalization of the input [182], such as graph [183, 184] and table [185, 186]. In addition to generating contents that adequately describes the input, several other challenges arise due to the structural organization of input data. The first challenge is the content selection, which primarily involves filtering out redundant information from the input. Another challenge is content planning, where the generation system should be capable of organizing the structure of predicted text [182]. It is worthwhile to explore the task of text generation with structured control codes.

6.2.2 Continual Learning for Incremental Control Codes

The preliminary chapters investigate various different control codes of text. To enforce the generation model to produce text that

satisfies diverse control codes, most existing approaches propose separate models for each text control factor [83, 187]. However, this might result in inference latency and large memory consumption during practical deployment. Although some efforts have been made to develop a single generation model capable of manipulating multiple styles simultaneously [188, 189], these approaches are limited to the control factors that have been explicitly provided before training. When new control factors are introduced, due to the catastrophic forgetting [190] of neural models, the generation model needs to undergo full parameter update. This often results in large consumption of computational resources. In this situation, a continual learning strategy that can support the control of emerging attributes while circumventing the multiple revisiting of previous training data is of great necessity.

□ **End of chapter.**

Publications During Ph.D. Study

1. **Jingjing Li**, Yifan Gao, Lidong Bing, Irwin King, Michael R. Lyu, *Improving Question Generation With to the Point Context*, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP 2019), pages 3216-3226, November 3-7, 2019.

2. **Jingjing Li**, Zichao Li, Lili Mou, Xin Jiang, Michael R. Lyu, and Irwin King. *Unsupervised Text Generation by Learning from Search*, in Advances in Neural Information Processing Systems (NeurIPS 2020), volume 33, pages 10820–10831, 2020.

3. Yifan Gao, Chien-Sheng Wu, **Jingjing Li**, Shafiq Joty, Steven C.H. Hoi, Caiming Xiong, Irwin King, Michael R. Lyu, *Discern: Discourse-Aware Entailment Reasoning Network for Conversational Machine Reading*, in Proceedings of the 2020

Conference on Empirical Methods in Natural Language Processing (EMNLP 2020), pages 2439-2449, November 16-20, 2020.

4. **Jingjing Li**, Zichao Li, Tao Ge, Irwin King, and Michael R. Lyu. *Text Revision by on-the-fly Representation Optimization*, in Proceedings of the 36_{th} AAAI Conference on Artificial Intelligence (AAAI 2022), volume 36, pages 10956–10964, June, 2022.

5. Xin Sun, Tao Ge, Shuming Ma, **Jingjing Li**, Furu Wei, and Houfeng Wang. *A Unified Strategy for Multilingual Grammatical Error Correction with Pre-trained Cross-lingual Language Model*, in Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22), pages 4367–4374, 2022.

6. Yueen Ma, Zixing Song, Xuming Hu, **Jingjing Li**, Yifei Zhang, Irwin King. *Graph Component Contrastive Learning for Concept Relatedness Estimation*, in Proceedings of the 37_{th} AAAI Conference on Artificial Intelligence (AAAI 2023), February 7-14, 2023.

Note: This thesis presents the work of paper [1, 2, 4].

Bibliography

[1] John Wieting and Kevin Gimpel. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*, 2017.

[2] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1123. URL https://www.aclweb.org/anthology/P17-1123.

[3] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. In *Proceedings of the 6th CCF Inter-*

national Conference on Natural Language Processing and Chinese Computing (NLPCC), pages 662–671, Dalian, China, November 2017.

[4] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997.

[5] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

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

[7] Tong Niu and Mohit Bansal. Polite dialogue generation without parallel data. *TACL*, 6:373–389, 2018.

[8] Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Compu-*

tational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 569–574, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2090.

[9] Emily Sheng, Kai-Wei Chang, P. Natarajan, and Nanyun Peng. Societal biases in language generation: Progress and challenges. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[10] Ruibo Liu, Chenyan Jia, Jason Wei, Guangxuan Xu, Lili Wang, and Soroush Vosoughi. Mitigating political bias in language models through reinforced calibration. In *AAAI Conference on Artificial Intelligence*, 2021.

[11] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *ArXiv*, abs/2009.11462, 2020.

[12] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2:

Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[13] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

[14] Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4912. URL https://aclanthology.org/W17-4912.

[15] Douglas Biber and Susan Conrad. *Register, genre, and style*. Cambridge University Press, 2019.

[16] Kate G Niederhoffer and James W Pennebaker. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21(4):337–360, 2002.

[17] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *ICML*, pages 1587–1596. PMLR, 2017.

[18] Hongyu Gong, Suma Bhat, Lingfei Wu, JinJun Xiong, and Wen-Mei Hwu. Reinforcement learning based text style transfer without parallel training corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3168–3180, 2019.

[19] Rui Zhang, Zhenyu Wang, Kai Yin, and Zhenhua Huang. Emotional text generation based on cross-domain sentiment transfer. *IEEE Access*, 7:100081–100089, 2019.

[20] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *NAACL-HLT*, pages 1865–1874, 2018.

[21] Sudha Rao and Joel R. Tetreault. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*, 2018.

[22] Cicero dos Santos, Igor Melnyk, and Inkit Padhi. Fighting offensive language on social media with unsupervised text style transfer. In *ACL*, pages 189–194, 2018.

[23] Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li, and Wen-Han Chao. Harnessing pre-trained neural networks with rules for formality style transfer. In *EMNLP/IJCNLP*, 2019.

[24] Xinyuan Zhang, Yi Yang, Siyang Yuan, Dinghan Shen, and Lawrence Carin. Syntax-infused variational autoencoder for text generation. In *ACL*, pages 2069–2078, 2019.

[25] Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. Syntax-guided controlled generation of paraphrases. *arXiv preprint arXiv:2005.08417*, 2020.

[26] Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. Training millions of personalized dialogue agents. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

[27] Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin.

Topic-guided variational auto-encoder for text generation. In *NAACL*, pages 166–177, 2019.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[29] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[30] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. 2014.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[32] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. *arXiv preprint arXiv:1809.02156*, 2018.

[33] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[34] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. *arXiv e-prints*, 2020.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[37] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa

Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[38] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[39] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *ArXiv*, abs/1409.3215, 2014.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[42] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[43] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL http://arxiv.org/abs/1802.05365.

[44] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf*, 2018.

[45] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina

Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[47] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[49] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[50] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and

Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

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

[52] Ruslan Mitkov and Le An Ha. Computer-aided generation of multiple-choice tests. *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003*, pages 15–, 2003.

[53] Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Christian Moldovan. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, 2010. URL https://www.aclweb.org/anthology/W10-4234.

[54] Michael Heilman and Noah A. Smith. Good question! sta-

tistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California, June 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N10-1086.

[55] Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2603. URL https://www.aclweb.org/anthology/W17-2603.

[56] Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. Improving neural question generation using answer separation. In *AAAI Conference on Artificial Intelligence*, 2019.

[57] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William

Cohen. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1040–1050, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1096. URL https://www.aclweb.org/anthology/P17-1096.

[58] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact aware neural abstractive summarization. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16121.

[59] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany, August 2016. Association

for Computational Linguistics. doi: 10.18653/v1/P16-1056. URL https://www.aclweb.org/anthology/P16-1056.

[60] Nitin Madnani and B. Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36:341–387, 2010.

[61] Kathleen R McKeown. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10, 1983.

[62] Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. *arXiv preprint cs/0304006*, 2003.

[63] Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual LSTM networks. In *COLING*, pages 2923–2934, 2016.

[64] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. In *ACL*, pages 3865–3878, 2017.

[65] Lihua Qian, Lin Qiu, Weinan Zhang, Xin Jiang, and Yong Yu. Exploring diverse expressions for paraphrase generation. In *EMNLP-IJCNLP*, pages 3164–3173, 2019.

[66] Qian Yang, Dinghan Shen, Yong Cheng, Wenlin Wang, Guoyin Wang, Lawrence Carin, et al. An end-to-end generative architecture for paraphrase generation. In *EMNLP-IJCNLP*, pages 3123–3133, 2019.

[67] Wanyu Du and Yangfeng Ji. An empirical comparison on imitation learning and reinforcement learning for paraphrase generation. In *EMNLP-IJCNLP*, pages 6014–6020, 2019.

[68] Shiqi Zhao, Haifeng Wang, Xiang Lan, and Ting Liu. Leveraging multiple MT engines for paraphrase generation. In *COLING*, pages 1326–1334, 2010.

[69] Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. Paraphrasing revisited with neural machine translation. In *ACL*, pages 881–893, 2017.

[70] Yinpeng Guo, Yi Liao, Xin Jiang, Qing Zhang, Yibo Zhang, and Qun Liu. Zero-shot paraphrase generation with multi-

lingual language models. *arXiv preprint arXiv:1911.03597*, 2019.

[71] Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. Generating sentences from disentangled syntactic and semantic spaces. In *ACL*, pages 6008–6019, 2019.

[72] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *AAAI*, volume 33, pages 6834–6842, 2019.

[73] AB Siddique, Samet Oymak, and Vagelis Hristidis. Unsupervised paraphrasing via deep reinforcement learning. In *SIGKDD*, pages 1800–1809, 2020.

[74] Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. Unsupervised paraphrasing by simulated annealing. *ArXiv*, abs/1909.03588, 2019.

[75] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep Learning for Text Style Transfer: A Survey. *Computational Linguistics*, 48(1):155–205, 04 2022.

[76] Aman Madaan, Amrith Rajagopal Setlur, Tanmay Parekh, Barnabás Póczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W. Black, and Shrimai Prabhumoye. Politeness transfer: A tag and generate approach. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

[77] Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. Paraphrasing for style. In *COLING*, pages 2899–2914, 2012.

[78] Jonas Belouadi and Steffen Eger. Bygpt5: End-to-end style-conditioned poetry generation with token-free language models. *arXiv preprint arXiv:2212.10474*, 2022.

[79] Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. On variational learning of controllable representations for text without supervision. In *ICML*, 2020.

[80] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[81] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi

Jaakkola. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6830–6841, 2017.

[82] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[83] Eric Malmi, Aliaksei Severyn, and Sascha Rothe. Unsupervised text style transfer with masked language models. In *EMNLP*, pages 8671–8680, 2020.

[84] Anubhav Jangra, Preksha Nema, and Aravindan Raghuveer. T-star: Truthful style transfer using amr graph as intermediate representation, 2022.

[85] Kaize Shi, Xueyao Sun, Li He, Dingxian Wang, Qing Li, and Guandong Xu. Amr-tst: Abstract meaning representation-based text style transfer. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4231–4243, 2023.

[86] Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. Imat: Unsupervised text attribute transfer

via iterative matching and translation. In *EMNLP-IJCNLP*, pages 3088–3100, 2019.

[87] Machel Reid and Victor Zhong. Lewis: Levenshtein editing for unsupervised text style transfer. *arXiv preprint arXiv:2105.08206*, 2021.

[88] cationR. Chandrasekar. Automatic induction of rules for text simpli. 1997.

[89] James E Hoard, Richard Wojcik, and Katherina Holzhauser. An automated grammar and style checker for writers of simplified english. In *Computers and Writing: State of the Art*, pages 278–296. Springer, 1992.

[90] S Rebecca Thomas and Sven Anderson. Wordnet-based lexical simplification of a document. In *KONVENS*, pages 80–88, 2012.

[91] Advaith Siddharthan. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4:77–109, 2006.

[92] Dan Feblowitz and David Kauchak. Sentence simplification as tree transduction. In *Proceedings of the second workshop*

on predicting and improving text readability for target reader populations, pages 1–10, 2013.

[93] Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. Text simplification using neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[94] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *TACL*, 4:401–415, 2016.

[95] Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. In *ACL*, pages 3393–3402, 2019.

[96] Shashi Narayan and Claire Gardent. Hybrid simplification using deep semantics and machine translation. In *ACL*, pages 435–445, 2014.

[97] Wei Xu, Chris Callison-Burch, and Courtney Napoles. Problems in current text simplification research: New data can help. *TACL*, 3:283–297, 2015.

[98] Mounica Maddela, Fernando Alva-Manchego, and Wei Xu. Controllable text simplification with explicit paraphrasing. *arXiv preprint arXiv:2010.11004*, 2020.

[99] Shashi Narayan and Claire Gardent. Unsupervised sentence simplification using deep semantics. In *INLG*, pages 111–120, 2015.

[100] Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. Unsupervised neural text simplification. *arXiv preprint arXiv:1810.07931*, 2018.

[101] Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7918–7928. Association for Computational Linguistics, 2020.

[102] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for

Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

[103] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-1013.

[104] Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the second workshop on statistical machine translation*, pages 228–231, 2007.

[105] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[106] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675, 2019.

[107] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT:

Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.704. URL https://aclanthology.org/2020.acl-main.704.

[108] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *CoRR*, abs/2106.11520, 2021. URL https://arxiv.org/abs/2106.11520.

[109] Hong Sun and Ming Zhou. Joint learning of a dual SMT system for paraphrase generation. In *ACL*, volume 2, pages 38–42, 2012.

[110] Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558, 2009.

[111] Ehud Reiter. A structured review of the validity of bleu. *Computational Linguistics*, 44(3):393–401, 2018.

[112] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[113] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[114] Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1802–1813, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1170. URL https://www.aclweb.org/anthology/P16-1170.

[115] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for

Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

[116] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

[117] Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1427.

[118] Swarnadeep Saha and Mausam. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguis-*

tics, pages 2288–2299, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1194.

[119] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1424.

[120] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099.

[121] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learn-

ing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL https://www.aclweb.org/anthology/P16-1154.

[122] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1014. URL https://www.aclweb.org/anthology/P16-1014.

[123] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3348. URL https://www.aclweb.org/anthology/W14-3348.

[124] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[125] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

[126] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[127] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceed-*

ings of the Thirteenth International Conference on Artificial Intelligence and Statistics, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

[128] Yifan Gao, Lidong Bing, Wang Chen, Michael R. Lyu, and Irwin King. Difficulty controllable generation of reading comprehension questions. In *Proceedings of the Twenty-Eightth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 8 2019.

[129] Yifan Gao, Piji Li, Irwin King, and Michael R. Lyu. Interconnected question generation with coreference alignment and conversation flow modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4853–4862, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-1480.

[130] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and

Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. In *ICLR*, 2018.

[131] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. In *EMNLP*, pages 5039–5049, 2018.

[132] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. In *NeurIPS*, pages 7057–7067, 2019.

[133] Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. Unsupervised neural text simplification. In *ACL*, pages 2058–2068, 2019.

[134] Eric Chu and Peter J Liu. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *ICML*, pages 1223–1232, 2018.

[135] Reinald Kim Amplayo and Mirella Lapata. Unsupervised opinion summarization with noising and denoising. In *ACL*, pages 1934–1945, 2020.

[136] Joseph 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.

[137] Susan W McRoy, Songsak Channarukul, and Syed S Ali. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420, 2003.

[138] Raphael Schumann, Lili Mou, Yao Lu, Olga Vechtomova, and Katja Markert. Discrete optimization for unsupervised sentence summarization with word-level extraction. In *ACL*, 2020.

[139] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8): 1771–1800, 2002.

[140] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.

[141] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, 1:1–20, 2010.

[142] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *ACL*, pages 425–430, 2015.

[143] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2017.

[144] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[145] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144, 2018.

[146] Hal Daumé III and Daniel Marcu. Learning as search opti-

mization: Approximate large margin methods for structured prediction. In *ICML*, pages 169–176, 2005.

[147] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to search better than your teacher. In *ICML*, 2015.

[148] Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, pages 1296–1306, 2016.

[149] Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In *ACL*, pages 4334–4343, 2019.

[150] Lifu Tu and Kevin Gimpel. Learning approximate inference networks for structured prediction. *arXiv preprint arXiv:1803.03376*, 2018.

[151] Lifu Tu and Kevin Gimpel. Benchmarking approximate inference methods for neural structured prediction. *arXiv preprint arXiv:1904.01138*, 2019.

[152] Yao Fu, Yansong Feng, and John P Cunningham. Paraphrase

generation with latent bag of words. In *NeurIPS*, pages 13623–13634, 2019.

[153] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. In *AAAI Conference on Artificial Intelligence*, 2017.

[154] Ruochen Xu, Tao Ge, and Furu Wei. Formality style transfer with hybrid textual annotations. *ArXiv*, abs/1903.06353, 2019.

[155] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *CoNLL*, pages 10–21, 2015.

[156] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*, 2018.

[157] Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*, 2018.

[158] Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Zhifang Sui, and Xu Sun. A dual reinforcement learning framework for unsupervised text style transfer. In *IJCAI*, pages 5116–5122, 2019.

[159] Kartik Goyal, Chris Dyer, and Taylor 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.

[160] Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *ACL*, pages 1651–1661, 2019.

[161] Xin Sun, Tao Ge, Shuming Ma, Jingjing Li, Furu Wei, and Houfeng Wang. A unified strategy for multilingual grammatical error correction with pre-trained cross-lingual language model. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4367–4374. International Joint Conferences on Artificial Intelli-

gence Organization, 7 2022. doi: 10.24963/ijcai.2022/606. URL https://doi.org/10.24963/ijcai.2022/606. Main Track.

[162] Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5057–5068, 2019.

[163] Tuhin Chakrabarty, Christopher Hidey, and Smaranda Muresan. Entrust: Argument reframing with language models and entailment. *arXiv preprint arXiv:2103.06758*, 2021.

[164] Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *ACL*, pages 4593–4601, 2019.

[165] Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. Perturbed masking: Parameter-free probing for analyzing and interpreting bert. In *ACL*, pages 4166–4176, 2020.

[166] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, volume 31, 2017.

[167] Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *SIGNLL*, pages 10–21, 2016.

[168] Xiang Lisa Li and Jason Eisner. Specializing word embeddings (for parsing) by information bottleneck. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2744–2754, 2019.

[169] Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. A multi-task approach for disentangling syntax and semantics in sentence representations. In *NAACL*, pages 2453–2464, 2019.

[170] Andrea Madotto, Zhaojiang Lin, Yejin Bang, and Pascale Fung. The adapter-bot: All-in-one controllable conversational model. *arXiv preprint arXiv:2008.12579*, 2020.

[171] Horacio Saggion. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137, 2017.

[172] Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. Neural crf model for sentence alignment in text simplification. *arXiv preprint arXiv:2005.02324*, 2020.

[173] Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, 2017.

[174] Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. Transforming complex sentences into a semantic hierarchy. In *ACL*, pages 3415–3427, 2019.

[175] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch, 1975.

[176] Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong

Zhang, Houfeng Wang, and Wenjie Li. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*, 2018.

[177] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[178] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Leveraging pre-trained checkpoints for sequence generation tasks. volume 8, pages 264–280. MIT Press, 2020.

[179] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP: System Demonstrations*, pages 38–45, Online, Oc-

tober 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[180] Jingjing Li, Yifan Gao, Lidong Bing, Irwin King, and Michael R. Lyu. Improving question generation with to the point context. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[181] Yifan Gao, Chien-Sheng Wu, Jingjing Li, Shafiq Joty, Steven CH Hoi, Caiming Xiong, Irwin King, and Michael R Lyu. Discern: Discourse-aware entailment reasoning network for conversational machine reading. *arXiv preprint arXiv:2010.01838*, 2020.

[182] Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *AAAI Conference on Artificial Intelligence*, 2018.

[183] Emilie Colin, Claire Gardent, Yassine Mrabet, Shashi Narayan, and Laura Perez-Beltrachini. The webnlg challenge: Generating text from dbpedia data. In *International Conference on Natural Language Generation*, 2016.

[184] Zhuoer Wang, Marcus Collins, Nikhita Vedula, Simone Filice,

Shervin Malmasi, and Oleg Rokhlenko. Faithful low-resource data-to-text generation through cycle training. *arXiv preprint arXiv:2305.14793*, 2023.

[185] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. Table-to-text generation by structure-aware seq2seq learning. 2018.

[186] Dragomir R. Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chia-Hsuan Hsieh, Nazneen Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zairi Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. Dart: Open-domain structured data record to text generation. *ArXiv*, abs/2007.02871, 2020.

[187] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[188] Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multiple-attribute text rewriting. 2018.

[189] Lei Sha and Thomas Lukasiewicz. Multi-type disentanglement without adversarial training. *ArXiv*, abs/2012.08883, 2020.

[190] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135, 1999.