# Knowledge-enhance Neural Text Generation

by

YE LIU
B.E., Northeastern University, 2015
M.S., University of Illinois at Chicago, 2016

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2021

Chicago, Illinois

Defense Committee:
Professor Philip S. Yu, Chair and Advisor
Professor Xinhua Zhang
Professor Natalie Parde
Professor Elena Zheleva
Professor Lifang He, Lehigh University

This dissertation is dedicated to my parents and my husband,

for their unconditional love and support.

## ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my Ph.D. advisor, Prof. Philip S. Yu, for his guidance and support throughout my Ph.D. study and research. It has been my privilege to work with you on different aspects of my Ph.D. journey. Your invaluable suggestions, guidance, and your passion for research not only help me with my past academic achievements but also will influence my professional career in the future.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Xinhua Zhang, Prof. Natalie Parde, Prof. Elena Zheleva, and Prof. Lifang He, for your valuable time serving as my dissertation committee members.

I am grateful to Prof. Lifang He at Lehigh University and Prof. Jiawei Zhang at the University of California San Diego, for their mentorship during my early research epoch and for enlightening me at the first glance of research. I would like to thank Dr. Chenwei Zhang and Prof. Yao Wan for invaluable suggestions and fruitful discussions during our collaborations in various research projects, which relate to this dissertation. My sincere thank goes to Dr. Kazuma Hashimoto, Dr. Yingbo Zhou, and Dr. Semih Yavuz, who mentored me when I was a research intern at Salesforce Research. Without your continuous support, this dissertation would not have been possible.

I would like to express gratitude to my fellow lab mates in the Big Data and Social Computing Lab at the University of Illinois at Chicago, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last

five years. My warmest thanks extend to all the collaborators, colleagues, and friends that I met at the University of Illinois at Chicago.

Last but not least, none of this could have happened without my family. I am grateful to my parents and grandparents and all other relatives, for their altruistic love and continuous encouragement. My greatest gratitude goes to my husband Fei Jiang, who has been taking perfect care of me. His unconditional love, support, and trust motivate me to work hard every day. His company is always my strongest weapon to fight against all difficulties and toughnesses.

YL

## CONTRIBUTIONS OF AUTHORS

Chapter 1 is an introduction that outlines my dissertation research.

Chapter 2 presents published papers (Liu et al., 2019) for which I was the primary author. Dr. Chenwei Zhang, Dr. Xiaohui Yan,Prof. Yi Chang and Prof. Philip S. Yu contributed to discussions with respect to the work and revising the manuscript.

Chapter 3 presents a published paper (Liu et al., 2020), for which I was the primary author. Dr. Yao Wan, Prof. Lifang He, Prof. Hao Peng, and Prof. Philip S. Yu contributed to discussions with respect to the work and revising the manuscript.

Chapter 4 presents a published paper (Liu et al., 2021) for which I was the primary author. Dr. Yao Wan, Dr. Jian-Guo Zhang, Dr. Wenting Zhao and Prof. Philip S. Yu contributed to discussions with respect to the work and revising the manuscript.

Chapter 5 presents a published paper (Liu et al., 2021) for which I was the primary author. Dr. Jian-Guo Zhang, Prof. Yao Wan, Dr. Congying Xia, Prof. Lifang He and Prof. Philip S. Yu contributed to discussions with respect to the work and revising the manuscript.

Chapter 6 concludes this dissertation.

## TABLE OF CONTENTS

## TABLE OF CONTENTS (Continued)

**CHAPTER** **PAGE**

## TABLE OF CONTENTS (Continued)

**CHAPTER** **PAGE**

## LIST OF TABLES

## LIST OF TABLES (Continued)

**TABLE** **PAGE**

## LIST OF FIGURES

## LIST OF FIGURES (Continued)

**FIGURE** **PAGE**

## SUMMARY

Text generation is one of the most important yet challenging tasks in natural language processing. Even though various neural generation models have been proposed to achieve the goal by generating output text from the input text, the input text alone usually provides limited knowledge to generate the desired output. Therefore, text generation performance is still far from satisfactory in many real-world scenarios. To make machines express like a human, we have considered incorporating various forms of internal and external knowledge beyond the input text into the generation models to solve this issue.

The first task is on the question refinement task, which incorporates deep reinforcement learning that considers both word-level rewards as immediate rewards but also question-level rewards as a long-time reward. For the second task, in light of the fact that the knowledge graph can provide the relational information to enhance the reasoning capacity and provide adjunct words to the concept, I propose the second approach. It is a novel knowledge graph-augmented framework for generative commonsense reasoning. In the third task, I incorporate the explicit syntactic and semantic structures of languages into a non-autoregressive Transformer, for the task of neural machine translation. In the fourth task, I'll introduce a Transformer-based pre-trained model with multi-granularity sparse attentions for long-text summarization.

# CHAPTER 1

# INTRODUCTION

Text generation is also known as natural language generation (NLG) informal. The goal of text generation is to generate the human language from various input formats such as textual data, image data, and knowledge graph. Among those different formats, text-to-text generation is the most popular-used and widely-studied task, which is what we study in this dissertation. Specifically, text generation converts the input text like a sequence or keywords to the semantic latent space vector, then processes this semantic latent space vector to the desired natural language. For instance, in machine translation system, it generates the same meaning sentence in different language based on the source input (Gu et al., 2018); in summarization system, it generates a summary of the source input, which includes the salient information (Li et al., 2020a); in question answering, it generates answers for the given input questions (Liu et al., 2020); in the dialogue system, it helps the chat-bots to communicate with the human (Mondal et al., 2018).

With the growth of deep learning models (Goodfellow et al., 2016; LeCun et al., 2015), neural text generation models have achieved remarkable performance in improving machines generation ability. A common workflow of the text generation model is under the sequence-to-sequence (seq2seq) format (Cho et al., 2014; Liu and Lapata, 2019b). Various text generation model uses this encoder-decoder schemes, for example, RNN (Cho et al., 2014; Sutskever et al., 2014), CNN (Gehring et al., 2017) and Transformer (Vaswani et al., 2017). This encoder-

decoder models have been widely-used in amount of text generation tasks such as machine translation (Gu et al., 2018), summarization (Li et al., 2020a; Huang et al., 2020) and question answering (Liu et al., 2019). Moreover, the attention mechanism (Vaswani et al., 2017) and copy mechanism (See et al., 2017) are two widely used methods to boost the performance of generation models.

## 1.1 Existing Works and Limitations

Since the input text only contains limited knowledge, it cannot support language generation models to produce the desired output. Therefore, the performance of generation is still far from satisfaction in many real-world scenarios. For example, in the question refinement task, given the ill-formed question: "What's the differncee between climate chnage and global wraming?". Conditioning on only the input text, a text generation system often produces meaningless and ill-grammar, such as "what's the difference between human beings and cancer cancer?"

Even in the current pre-trained language model, such as GPTs (Radford et al., 2019; Brown et al., 2020), UniLM (Dong et al., 2019), T5 (Raffel et al., 2020) and BART (Lewis et al., 2020), although they can capture rich language information from text sentence corpus and generate accurate language texts, almost all of them ignore knowledge information and thereby fail to generate output towards capturing the human commonsense. For example, given a set of commonsense concepts {river, fish, net, catch}, the task is to generate a coherent sentence describing a scenario covering all given concepts, such as "Fisherman uses a strong net to catch plentiful fishes in the river". From our analysis, we note that the state-of-the-art pre-trained models generate implausible and anomalous sentences in this task - e.g., GPT-2 generated "A

fish is catching in a net", UniLM generated "A net catches fish", etc. Moreover, the generated sentences by the pre-trained models are simple and rigid, while the human sentence is more natural and rich, like "plentiful fishes", "wide river", etc.

Human beings generate language in a different manner. They constantly acquire, understand, and store knowledge from their daily life, which comes from broader sources. Therefore, their learned knowledge can help them to communicate, read and write in the new situation. For example, in question answering, people could use the commonsense or professional knowledge they learned to answer the question; in summarization, people could acquire the keywords or important information from the long document and organize them into understandable content to respond; in communication like a chatbot, people could fluently communicate with theirs using their background knowledge. Therefore, if we can help machines contain that knowledge that is beyond the input sequence, the text generation can achieve much better performance.

## 1.2 What is Knowledge-enhanced Text Generation?

Generally speaking, knowledge is a familiarity, awareness, or understanding of someone or something (Horibe, 1999). In NLG systems, knowledge is an awareness and understanding of the input text and its surrounding context. We separate the knowledge into two kinds namely, internal knowledge and external knowledge. In internal knowledge, it is about extracting the knowledge from the given input text aiming to better understand its surrounding contexts, such as linguistic features (part-of-speech and name entity recognition), keywords, or topics. And in the external knowledge, it represents obtaining the knowledge from the external knowledge source like open-domain knowledge graph ConceptNet (Speer et al., 2017) and textual

documents corpus like Wikipedia. The research works on incorporating knowledge into text generation is called knowledge-enhanced text generation.

**Knowledge-enhanced Text Generation**. In the text generation task, given the input sequence $\mathbf{X}$, the neural text generation model aims to generate the output sequence $\mathbf{Y}$. Assume we have already access to additional related knowledge represented as $\mathbf{K}$. This knowledge $\mathbf{K}$ can either be obtained from external knowledge like open-domain sources or from internal knowledge extraction. Moreover, this knowledge $\mathbf{K}$ can either be acquired from input sequence $\mathbf{X}$ or output sequence $\mathbf{Y}$. Knowledge-enhanced text generation aims to incorporate the knowledge $\mathbf{K}$ to enhance the quality of the generated output sequence $\mathbf{Y}$ given the input sequence $\mathbf{X}$. In this dissertation, we will answer two questions that commonly appear in knowledge-enhanced text generation in each work, which are how to acquire knowledge and how to incorporate knowledge to facilitate text generation.

## 1.3 Enhance Text Generation with External Feedback

(Part of this chapter was previously published in (Liu et al., 2019))

In real-world question-answering (QA) systems, ill-formed questions, such as wrong words, ill word order, and noisy expressions, are common and may prevent the QA systems from understanding and answering them accurately. In order to eliminate the effect of ill-formed questions, we approach the question refinement task and propose a unified model, QREFINE, to refine the ill-formed questions into well-formed questions. The basic idea is to learn a Seq2Seq model to generate a new question from the original one. To improve the quality and retrieval performance of the generated questions, we make two major improvements: 1) To

better encode the semantics of ill-formed questions, we enrich the representation of questions with character embedding and the recent proposed contextual word embedding such as BERT, besides the traditional context-free word embeddings; 2) To make it capable to generate desired questions, we train the model with deep reinforcement learning techniques that considers an appropriate wording of the generation as an immediate reward and the correlation between generated question and answer as time-delayed long-term rewards. Experimental results on real-world datasets show that the proposed QREFINE method can generate refined questions with more readability but fewer mistakes than the original questions provided by users. Moreover, the refined questions also significantly improve the accuracy of answer retrieval.

## 1.4 Enhance Text Generation with Knowledge Graph

(Part of this chapter was previously published in (Liu et al., 2020))

Generative commonsense reasoning which aims to empower machines to generate sentences with the capacity of reasoning over a set of concepts is a critical bottleneck for text generation. Even the state-of-the-art pre-trained language generation models struggle at this task and often produce implausible and anomalous sentences. One reason is that they rarely consider incorporating the knowledge graph which can provide rich relational information among the commonsense concepts. To promote the ability of commonsense reasoning for text generation, we propose a novel knowledge graph-augmented pre-trained language generation model KG-BART, which encompasses the complex relations of concepts through the knowledge graph and produces more logical and natural sentences as output. Moreover, KG-BART can leverage the graph attention to aggregate the rich concept semantics that enhances the model gener-

alization on unseen concept sets. Experiments on benchmark CommonGen dataset verify the effectiveness of our proposed approach by comparing with several strong pre-trained language generation models, particularly KG-BART outperforms BART by 5.80, 4.60, in terms of BLEU-3, 4. Moreover, we also show that the generated context by our model can work as background scenarios to benefit downstream commonsense QA tasks.

## 1.5 Enhance Text Generation with Syntactic and Semantic Structure

(Part of this chapter was previously published in (Liu et al., 2021))

The non-autoregressive models have boosted the efficiency of neural machine translation through parallelized decoding at the cost of effectiveness when compared with the autoregressive counterparts. In this paper, we claim that the syntactic and semantic structures among natural language are critical for non-autoregressive machine translation and can further improve performance. However, these structures are rarely considered in existing non-autoregressive models. Inspired by this intuition, we propose to incorporate the explicit syntactic and semantic structures of languages into a non-autoregressive Transformer, for the task of neural machine translation. Moreover, we also consider the intermediate latent alignment within target sentences to better learn the long-term token dependencies. Experimental results on two real-world datasets (i.e., WMT14 En-De and WMT16 En-Ro) show that our model achieves a significantly faster speed, as well as keeps the translation quality when compared with several state-of-the-art non-autoregressive models.

## 1.6 Enhance Text Generation with Graph Learning

(Part of this chapter was previously published in (Liu et al., 2021))

To capture the semantic graph structure from raw text, most existing summarization approaches are built on GNNs with a pre-trained model. However, these methods suffer from cumbersome procedures and inefficient computations for long-text documents. To mitigate these issues, this paper proposes HETFORMER, a Transformer-based pre-trained model with multi-granularity sparse attentions for long-text extractive summarization. Specifically, we model different types of semantic nodes in the raw text as a potential heterogeneous graph and directly learn heterogeneous relationships (edges) among nodes by Transformer. Extensive experiments on both single- and multi-document summarization tasks show that HETFORMER achieves state-of-the-art performance in Rouge F1 while using less memory and fewer parameters.

# CHAPTER 2

# ENHANCE TEXT GENERATION WITH EXTERNAL KNOWLEDGE

This chapter was previously published as "Generative question refinement with deep reinforcement learning in retrieval-based QA system" in CIKM'18 (Liu et al., 2019). DOI:https://doi.org/10.1145/3357384.3358046.

## 2.1 Introduction

QA systems greatly facilitate the information access of users, which are popularly used in both Web and mobile Internet. In these systems, users input a question either by text or voice and expect to get the right answer quickly. However, due to factors such as thoughtless questions from users, misoperations of keyboard input and ASR (automatic speech recognizer) error, the ill-formed questions asked by users are usually expressed with vagueness, ambiguity, noises and errors.

By manual analysis on the WikiAnswer dataset [1], we find that about 68% questions are ill-formed. As shown in Table I, there are three typical ill-formed question types, specifically, wrong words, ill words order and noisy background, and they include 79%, i.e., ((21% + 23% + 12%)/68%) ill-formed questions. Generally, a question is a short sentence with a few words in QA systems. Directly using ill-formed questions to search for answers in a retrieval based QA

[1] http://knowitall.cs.washington.edu/oqa/data/WikiAnswers

systems (Faruqui and Das, 2018) will hurt the downstream steps, e.g., answer selection (Tan et al., 2015) and hence compromise QA systems' effectiveness.

Inspired by the task of query refinement in web search (Nogueira et al., 2018), we study the task of question refinement in QA system, which aims to improve the quality of users' questions, in the meanwhile, boost the accuracy of the downstream answer retrieval. We can see that the task is complex since it contains the following subtasks: 1) word correction, e.g., correct *"defenition"* to "definition"; 2) word reorder, e.g., ill words order example in Table I; 3) sentence simplification, e.g., remove the redundant expression like "based on tiresias prediction" in noisy background example in Table I.

| **Type of Ill-formulation** | **Ratio in Data** | **Ill-formed Question** | **Well-formed Questions** |
|---|---|---|---|
| Wrong words | 21% | what is the *defenition* of the word infer | what is the definition for inference |
| Ill words order | 23% | limestone is what kind of *rocke* | what is limestone rock |
| Noisy background | 12% | based on tiresias prediction which heroic qualities will odysseus need to rely upon as he continues his journey | what heroic qualities does odysseus rely on |

TABLE I. Examples of the three common types of ill-formed and generated well-formed questions on WikiAnswers dataset. The ratio in data is counted within 1000 random sampled triples. The other ill-formed questions belong to several other types which have a minority percentage.

An intuitive way is to tackle these problems one by one alone. For instance, Xie et.al (Xie et al., 2016) proposed a charater-level text correction to deal with the orthographic errors and

rare words. Yuan et.al (Yuan and Briscoe, 2016) focus on grammar error correction to correct the erroneous word phrases. Besides, Zhang et.al. (Zhang and Lapata, 2017) utilized deep reinforcement learning to simplify questions, like splitting complex questions and substitutes difficult words with common paraphrases. However, it's laboursome to combine these methods together in practice, which might require no domain knowledge and a few human intervention.

Is it possible to tackle these problems with a unified model? Inspired by the successful usage of sequence-to-sequence (Seq2Seq) model (Sutskever et al., 2014) on related tasks such as machine translation (Britz et al., 2017), text summarization (Nallapati et al., 2016), and sentence simplification (Zhang and Lapata, 2017), it is promising to use it in the question refinement task. Seq2Seq model is flexible enough to encode patterns for sequence transformation such as word correction, word recorder, and sentence simplification, if there are appropriate training datasets. Unfortunately, we find that the vanilla Seq2Seq model does not perform well on this task. The reasons may be twofold: 1) it fails to learn a good representation of ill-formed questions, which might contain many wrong or noisy words. 2) The maximize likelihood objective is not consistent with our target, i.e., generated better quality questions and thus improve the accuracy of answer retrieval.

To overcome these problems, we develop a Seq2Seq model for question refinement called QREFINE. For the question representation, since a well-formed question might sensitive to the word order, we make use of the recent proposed contextual word embeddings such as BERT (Devlin et al., 2018) to capture the contextual word information. As BERT is trained over a large scale unlabeled corpus, it also can alleviate the data sparsity problem where there is

not enough training data. Moreover, considering the ill-formed questions might contain typos, we also incorporating the fine-grained character embedding (Pan et al., 2017) as a part of question representation. Our experimental results show that the two types of representations substantially improve the effectiveness of the Seq2Seq model.

To make the Seq2Seq model generate desired questions, we develop a training algorithm based on reinforcement learning. we assign not only word-level rewards to each word for its wording from a pertained language model and Bert language model as immediate rewards but also question-level rewards such as the correlation of the refined question to its answer. In order to solve the low data efficiency and unrobust policy problems on the traditional policy gradient method, we use advanced policy gradient method proximal policy optimization (PPO) (Tuan et al., 2018) for well-formed question generation (Schulman et al., 2017; Tuan et al., 2018). We compared our model with the state-of-the-art baselines in two QA datasets. The result shows our model outperforms baselines on question refinement. Besides, the case studies show the improved readability of the questions after refinement using QREFINE, and its effectiveness in improving the utility of an existing QA system. Moreover, it's worth to notice that our model is fully data-driven and might not require domain knowledge and human intervention.

## 2.2 Preliminary

We formally define the question refinement task studied in this paper. After that, we introduce some terminologies that we will use throughout the paper.

### 2.2.1 Problem Description

Given an ill-formed question consists of $\mathbf{x} = [x_1, x_2, ..., x_N]$ of an arbitrary-length $N$, the well-formed question $\mathbf{y} = [y_1, y_2, ..., y_M]$ of a variable-length $M$. The aim of question refinement is to refine $\mathbf{x}$ to $\mathbf{y}$ which has better readability. It is expected that the generated well-formed question $\mathbf{y}$ can be better able to retrieve the best answer candidate $\mathbf{a_k} = [a_1, a_2, ..., a_L]$, where $1 \leq k \leq s$ from an answer candidate pool $\{\mathbf{a}_1, \mathbf{a}_2, ..., , \mathbf{a}_s\}$.

### 2.2.2 Seq2Seq Framework on Question Refinement

The Seq2Seq model adopts an encoder-decoder framework that learns to encode an ill-formed question $\mathbf{x}$ into a fixed-length latent semantic representation and to decode the latent semantic representation back into a variable-length well-formed question $\mathbf{y}$. Formally, Seq2Seq model is a general method that learns the conditional distribution over a variable-length sequence conditioned on the other variable-length sequence, namely, $p_{lm}(y_1, ..., y_M | x_1, ..., x_N)$.

The encoder can be a convolution neural network or a recurrent neural network that summarizes the ill-formed question into a vector representation. Since LSTM (Hochreiter and Schmidhuber, 1997) is good at learning long-term dependencies in the data (Graves, 2013), we adopt LSTM to sequentially encode each word of ill-formed question $\mathbf{x}$. As the LSTM reads each word, the hidden state of the LSTM is updated $h_n = \text{LSTM}_{\text{encoder}}(h_{n-1}, x_n)$. Therefore, the encoder transforms the ill-formed question $\mathbf{x}$ into a sequence of hidden states $(h_1, h_2, ..., h_N)$.

The decoder can be another LSTM which is trained to generate the current hidden state $k_m$ based on the current word $y_m$ and the previous hidden state $k_{m-1}$:

$$k_m = \text{LSTM}_{\text{decoder}}(k_{m-1}, y_m). \tag{2.1}$$

Moreover, as introduced in (Luong et al., 2015), a context vector $c_m$ can be obtained for each decoder step $m$ by being attentive to the encoding of the source question dynamically:

$$c_m = \sum_{n=1}^{N} \alpha_{nm} h_n, \tag{2.2}$$

where $c_m$ is a weighted sum of the hidden states of the ill-formed question $\mathbf{x}$: The attention score $\alpha_{nm}$ between the $n$-th ill-formed question hidden unit $h_n$ and $m$-th well-formed question hidden unit $k_m$ is calculated as follows:

$$\alpha_{nm} = \frac{exp(h_n^T \cdot k_m)}{\sum_{l=1}^{N} exp(h_l^T \cdot k_m)}. \tag{2.3}$$

Formally, the Seq2Seq model is formed as below:

$$p_{lm}(y_m | y_{1:m-1}, \mathbf{x}) = \text{softmax}(g(k_m^T, c_m)), \tag{2.4}$$

where $g(\cdot)$ is an activation function $g(k_m^T, c_m) = \mathbf{W}_o \text{tanh}(\mathbf{U}_h k_m^T + \mathbf{W}_h c_m)$, where $\mathbf{W}_o \in \mathbb{R}^{|V| \times d}, \mathbf{U}_h \in \mathbb{R}^{d \times d}$ and $\mathbf{W}_h \in \mathbb{R}^{d \times d}$; $|V|$ is the output vocabulary size and $d$ is the hidden unit size.

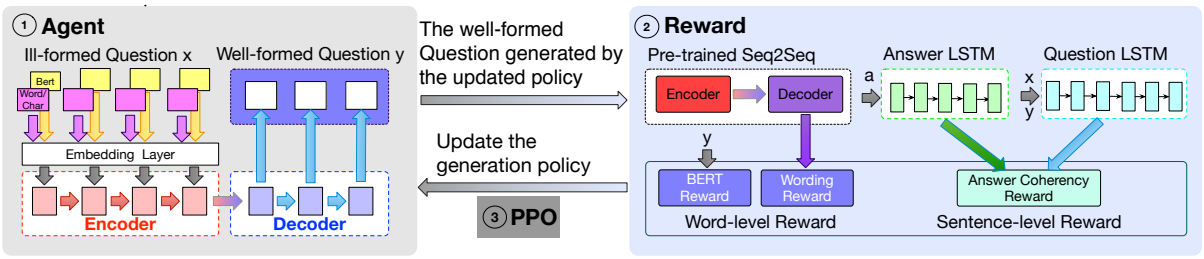



Figure 1. The architecture of the proposed model QREFINE. ① The encoder of the agent module reads the ill-formed question and the decoder generates a well-formed question, one word/phrase at a time. ② The well-formed question being generated so far is sent to a pre-trained reward module, which calculates a word-level wording reward from word-level LM and BERT Reward and a question-level answer correlation reward from QA similarity. ③ The PPO module updates agent's generation policy, aiming to maximize the rewards.

## 2.3 Reinforced Generative Question Refinement

### 2.3.1 Model Description

Despite the successful application in numerous sequence transduction tasks (Bahdanau et al., 2014), a vanilla Seq2Seq model is not ideal for question refinement since it only makes a few trivial changes of ill-formed question (Zhang and Lapata, 2017). To encourage a wider variety of rewrite operations while keeping the refined question fluent and coherent to the answer, we employ a reinforcement learning framework (see Figure 1).

The refinement agent first reads the ill-formed question $\mathbf{x}$ from the encoder; and then at each step of the decoder, it takes an action $y_m \in V$, where $V$ is the output vocabulary, according to a policy $\pi_\theta(y_m|y_{1:m-1}, \mathbf{x})$. The agent continues to take actions until it produces <EOS> (denoting end of sentence) token yielding the generated well-formed question of our model

$\mathbf{y} = [y_1, y_2, ..., y_M]$. Two types of rewards, wording reward, and answer correlation reward, are received and the advanced policy gradient method PPO is used to update the agent. In the following, we introduce our question representation and reward function. After that, we present the details of the processes for generating accurate and consistent questions by leveraging the REINFORCE and PPO method.

### 2.3.2 Question Representation

The ill-formed question has a wrong semantic order and contains some unrelated background information. To solve the problem, the model need to learn the institutional utterance of words, which needs to consider the correlation between words and words. The widely-used context-free models such as Skip-gram (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) cannot consider the correlation between words. Because they produce a unique word embedding for each word in the vocabulary, so that "apple" would have the same hidden representation in "red apple" and "apple store". However contextual models, like BERT can generate a representation of each word based on the other words in the sentence. Therefore, we concatenate the context-free and contextual embedding together as the word embedding to capture such coarse-grained correlation patterns of words.

Since the ill-formed question always contains the misspelled words, which is usually set as <UNK>, which is hard to capture the meaning of the original word. To capture the meaning of the misspelled words, we extend the word expression by incorporating fine-grained character expression. By using the character-level embedding, we can get the high-dimensional vector representation. As the character-level input, the original sentence is broken up to a sequence of

characters, which includes special characters, such as quotation mark. Characters are embedded into vectors, which can be considered as 1D inputs to the Bidirectional Long Short-term Memory Network (BI-LSTM) (Hochreiter and Schmidhuber, 1997), and the hidden layer of the last LSTM unit is the fixed-size vector of each word. Overall, we combine the fine-grained character-level embedding and coarse-grained contextual and context-free word embedding to represent the question.

### 2.3.3 Reward

The reward for system output $\mathbf{y}$ is the weighted sum of two types of rewards aimed at achieving well-formed readable and answer correlation question: wording rewards on the word-level from the Reward RNN and BERT, which aims to measure how well each generated word is in line with the language model (LM) rule, and the question-level answer correlation reward that has the ability to infer the correlation of the refined question to its answer, even if it is not generating until the end of the well-formed question.

**Wording Reward** The wording reward $r_w$ aims to give an immediate reward to each of words when it is being generated in the well-formed question. BERT pre-trained on the large dataset like Wikipedia could give the contextual wording reward $r_B(y_t) = p_B(y_t|y_{1,\ldots,M})$, which is the probability of word $y_t$ given by BERT model. Moreover, for the domain-specific, we also use the decoder of the pre-trained Seq2Seq module as a trained LM Reward RNN which is able

to score the probability of the next word given the words generated so far. Thus, the wording reward of the $t$-th word in the well-formed question is:

$$r_w(y_t) = r_B(y_t) + p_{lm}(y_{t+1}|k_t), \tag{2.5}$$

where $k_t$ is the current state which is the hidden representation of the generated well-formed question with $t$ words so far: $[y_1, y_2, ..., y_t]$ and $y_{t+1}$ is the generated word in the $(t+1)$-step.

**Answer Correlation Reward** The refinement result should not only improve the readability of the question, more importantly, have better ability to address its correlation to the answer once refined. With this motivation, we design an answer correlation reward $r_{ac}$ to further measure the correlation of the refined question to its answer on the question-level as a whole. As answers themselves are sometimes ill-formed and contain a large amount of unrelated information, they may not share lexical units with the well-formed question directly. But the well-formed question is semantically easier to answer than the ill-formed question. Following the similar ranking loss in (Tan et al., 2015; Feng et al., 2015), the answer correlation module defines the training objective as a hinge loss:

$$r_{ac}(\mathbf{y}) = max\{0, \quad \epsilon - \text{sim}(\text{LSTM}_q(\mathbf{x}),\ \text{LSTM}_a(\mathbf{a})) + \text{sim}(\text{LSTM}_q(\mathbf{y}),\ \text{LSTM}_a(\mathbf{a}))\}, \tag{2.6}$$

where we use two separate LSTMs, $\text{LSTM}_q$ and $\text{LSTM}_a$, to encode the question and answer to the vector representation. The well-formed question and ill-formed question share the same $\text{LSTM}_q$ , and $\epsilon$ is the constant margin. Furthermore, $\text{sim}(\text{LSTM}_q(\mathbf{x}), \text{LSTM}_a(\mathbf{a})) = \text{LSTM}_q(\mathbf{x})\mathbf{W}_{\text{sim}}\text{LSTM}_a(\mathbf{a})^T$

computes a bi-linear term between the question and its correct answer. We train the model by maximizing answer correlation reward $r_{ac}$ using ground-truth well-formed and ill-formed questions to learn the weight of $\text{LSTM}_q$ , $\text{LSTM}_a$ network and $\mathbf{W}_{\text{sim}}$. After that, we hold a fixed copy of networks to give rewards to the generated well-formed question.

**Accumulated Reward** We add the answer correlation reward to the end of the wording reward, as the overall evaluation of the generated question. The QREFINE reward $r$ of each word is the combination of the wording reward and the answer correlation reward,

$$r(y_i) = \begin{cases} r_w(y_i), i \neq M \\ r_w(y_i) + c_1 r_{ac}(\mathbf{y}), i = M \end{cases} \tag{2.7}$$

where $c_1$ is the parameter to tune the weight between wording reward $r_{lm}$ and answer correlation reward $r_{ac}$; Since we want QREFINE module to have the ability to infer reward even if not reaching the end of the generation process and the future reward will influence the current reward, we adopt the accumulated QREFINE reward $R$ with the discounted factor $\gamma$ and the accumulated QREFINE discounted reward of $t$-th word is represented as,

$$R(y_t) = \gamma^0 r(y_t) + \gamma^1 r(y_{t+1}) + \cdots + \gamma^{M-t} r(y_M). \tag{2.8}$$

By using the accumulated reward, we are able to infer the answer correlation reward even if we are not reaching the end of the generation process.

### 2.3.4 Question Generation

A popular choice of loss in traditional models is the cross-entropy used to maximize the probability of the next correct word. However, this loss is at the word-level and the performance of these models is typically evaluated using discrete metrics. To overcome, we employ the intuitions of deep reinforcement learning, which incorporates policy exploration and exploitation into the word generation. Compared to the traditional word generation way that learns a sequential model to greedily generate the next correct target word, we take advantage of the policy network with a reward function to jointly discover the next best word at each time step, aiming to maximize the reward of whole sentence.

Question refinement can be formulated as a Markov decision process (MDP) $(S, A, P, R)$, where $S$ denotes a set of states $s_t = \{y_{1:t-1}, \mathbf{x}\}$, $A$ represents a set of actions $a_t = y_t$, $P$ is the state transition probability of the next state given the current state and action, $R$ is a reward function $r(s_t, a_t)$ for every intermediate time step $t$, and $\gamma$ is a discount factor that $\gamma \in [0, 1]$.

The actions are taken from a probability distribution called policy $\pi$ given the current state (i.e., $a_t \sim \pi(s_t)$). In question refinement, $\pi$ is a seq2seq model. Therefore, reinforcement learning methods are suitable to apply to question refinement model by learning the seq2seq model, or policy $\pi$, that can gain reward as much as possible.

**On-policy Optimization** Due to the high dimensional action space for question refinement and high diversity of the required generation result, policy gradient method, like REINFORCE (Barto and Sutton, 1998) are more appropriate in the question generation than value-based methods like Q-learning (Mnih et al., 2013).

For a given ill-formed question $\mathbf{x}$, we want to return a formulated question $\mathbf{y}$, maximizing an accumulated reward $R$. The answer $\mathbf{a}$ is the known given by the database. The question $\mathbf{y} \sim \pi_\theta(\cdot|\mathbf{x})$ is generated according to $\pi_\theta$ where $\theta$ is the policy's parameter and the goal is to maximize the expected reward of the reformulated question under the policy, $E_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[R(\mathbf{y})]$.

Given reward $r_t$ at each time step $t$, the parameter $\pi$ of policy $\pi$ (a seq2seq model) is updated by policy gradient as follows:

$$E_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[R(\mathbf{y})] \approx \frac{1}{N}\sum_{i=1}^{N} R(y_i), y_i \sim \pi_\theta(\cdot|\mathbf{x}). \tag{2.9}$$

To compute gradients for training we use REINFORCE(Williams and Peng, 1991),

$$\begin{aligned} &\nabla E_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[R(\mathbf{y})] \\ &= E_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}\nabla_\theta log(\pi_\theta(\mathbf{y}|\mathbf{x}))R(\mathbf{y}) \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta log(\pi_\theta(y_i|y_{1:i-1}, X))R(y_i), y_i \sim \pi_\theta(\cdot|\mathbf{x}). \end{aligned} \tag{2.10}$$

The above reward estimator often leads to the problem of high variance, which results in unstable training (Greensmith et al., 2004). To overcome this problem, we let the estimator to minus a baseline: $B(\mathbf{x}) = E_{\mathbf{y}\sim\pi_\theta(\cdot|\mathbf{x})}[R(\mathbf{y})]$ (Sutton et al., 2000). This expectation is also computed by sampling from the policy given $\mathbf{x}$.

To avoid the model not being able to explore new words that could lead to a better answer correlation question, we use entropy regularization:

$$\mathbf{H}[\pi_\theta(\mathbf{y}|\mathbf{x})] = \sum_{i=1}^{N} \sum_{y_i \in V} log(\pi_\theta(y_i|y_{1:i-1}, \mathbf{x}))\pi_\theta(y_i|y_{1:i-1}, \mathbf{x}) \tag{2.11}$$

The final objective is:

$$\mathbb{E}_{\mathbf{y} \sim \pi_\theta(\cdot|\mathbf{x})}[R(\mathbf{y}) - B(\mathbf{x})] + \lambda \mathbf{H}[\pi_\theta(\mathbf{y}|\mathbf{x})], \tag{2.12}$$

where $\lambda$ is the regularization weight. $R(\mathbf{y}) - B(\mathbf{x})$ can be interpreted as the goodness of adopted action at over all the possible actions at state st. Policy gradient directly updates $\pi$ to increase the probability of at given $s_t$ when advantage function is positive, and vice versa.

**Off-policy Optimization** The vanilla policy gradient method is on-policy method and have convergence problem (Schulman et al., 2015). Empirically they often lead to problems like low data efficiency and unreliable performance, as shown in subsection 2.4.6. We use the advanced deep reinforce method proximal policy optimization (PPO) (Schulman et al., 2017) to learn a more stable policy.

Proximal policy optimization (PPO) (Tuan et al., 2018) is modified from trust region policy optimization (TRPO) (Schulman et al., 2015), and both methods aim to maximize a surrogate objective and subject to a constraint on quantity of policy update:

$$max_\theta L^{TRPO}(\theta) = \mathbb{E}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}(a_t|s_t)}} A_t], \tag{2.13}$$

$$\text{subject to } \mathbb{E}[\mathbf{KL}[\pi_{\theta_{old}(a_t|s_t)}, \pi_{\theta(a_t|s_t)}]] \leq \delta$$

$\pi_{old}$ is the old parameters before update. Because the KL-divergence between $\pi$ and $\pi_{old}$ is bounded by $\delta$, the updated policy $\pi$ cannot be too far away from the old policy $\pi_{old}$.

To optimize policy, PPO alternates between sampling sentence generated from the current policy and performing several epochs of optimization on the sampled sentences. According to the paper (Schulman et al., 2017), the clipped objective to heuristically constrain the KL-divergence setting achieves the best performance:

$$L_t^{CLIP}(\theta) = \mathbb{E}_t[min(r_t(\theta)clip(r_t(\theta), 1-\epsilon, 1+\epsilon))\hat{A}_t], \tag{2.14}$$

where $\beta_t$ denotes the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ and $\epsilon$ is a hyperparameter (e.g., $\epsilon = 0.1$). When $\hat{A}_t$ is positive, the objective is clipped by $(1+\epsilon)$; when $\hat{A}_t$ is negative, the objective is clipped by $(1-\epsilon)$.

$\hat{A}_t$ is the expected advantage function (the expected rewards minus a baseline like $V(k_t)$) which can be calculated as:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{2.15}$$

$$\delta_t = r_t + \gamma V(k_{t+1}) - V(k_t).$$

To improve the exploration of our model for generating diverse yet coherent words that could constitute a better well-formed question, we use entropy regularization. The integrated PPO method is shown as below:

$$L_t^{PPO}(\theta) = \mathbb{E}_t[L_t^{CLIP}(\theta) + c_2\pi_\theta(y_{t+1}|k_t)log(\pi_\theta(y_{t+1}|k_t))]. \tag{2.16}$$

The algorithm of QREFINE with PPO optimization shows in Alg. 1

**Learning** The training stage of traditional models suffer from the exposure bias (Ranzato et al., 2016), which is caused by the ground-truth words are missing in the testing time and previously decoder generated words are applied to predict the distribution of the next word. In the testing phase, this exposure bias makes error accumulated and makes these models become sub-optimal. Therefore the seq2seq model is not capable of generating the words which are appropriate but having low probability to be selected in the testing stage.

In order to solve the exposure bias problem, we train the model by using MIXER algorithm described in (Ranzato et al., 2016) to expose both training data and its predictions. In the inference stage, we greedily selected the word that has the highest probability to generate the question stopping until the <EOS> token is generated.

## 2.4 Experiments

In this section, we evaluate the proposed methods on two real-world datasets, by examining the readability of the refined questions both quantitatively and qualitatively. A question-answer

**Algorithm 1** QREFINE-PPO

**Input:** Ill-formed question $X$, Well-formed question $Y$, rating data $R$, the number of episodes $K$, $\varepsilon$-greedy parameter $\varepsilon$, ratio of language model reward and answer correlation reward $c_1$, the discounted factor of RL $\lambda$, the threshold $\epsilon$ and the entropy regularization $c_2$

**Output:** the learned policy $p_\theta$

Initialize policy $p_\theta$ and old policy $p_{\theta_{old}}$ with supervised pretrained policy $p_{\theta'}$
**for** episode = 1, ..., K **do**
  Uniformly pick a batch of ill-formed question $u \in \mathcal{U}_{train}$ as the environment
  Start to generate the word according to $p_{\theta_{old}}(y_i|X)$ until the <EOS> token is generated, the generated sentence as $Y'$
  Send $X$ and $Y'$ to the BERT mechine and pretrained word embedding model to calculate the word-level reward
  Send $X$, $Y$ and $Y'$ to the qa-lstm model to calculate the sentence-level reward, based on Equation 2.6
  Calculate the advantage function of each time step according to Equation 2.15
  **repeat**
    Update the policy $p_\theta$ using Equation 2.16
  **until** convergence
  Set old policy $p_{\theta_{old}}$ to policy $p_\theta$
**end for**

retrieval experiment is used to evaluate the performance of the generated question. In the end, we further conduct the ablation study and learning curve of the method.

### 2.4.1 Dataset

The datasets are formatted as triples, where each instance consists of a well-formed question, an ill-formed question and an answer in each triple.

**Yahoo**: The Yahoo non-factoid question dataset [1] is collected from Yahoo Webscope that the questions would contain non-factoid answers in English. After limiting the length of the

[1] https://ciir.cs.umass.edu/downloads/nfL6/

answer, we have 85k questions and their corresponding answers. Each question contains its best answer along with additional other answers submitted by users. The average length of ill-formed questions is around 12 tokens and the average length of well-formed questions is around 10 tokens, with 73k vocabulary size and 44 different characters. The average length of answers is around 39 tokens.

To testify the refinement performance from ill-formed to well-formed question, we generate the ill-formed question on three types of the ill-formed question, wrong words, wrong order and Noisy background. We randomly change the character of the words or change the order of the character of words to generate the **Wrong Word** dataset. For the **Wrong Order** dataset, we randomly change the order of some fragments in the well-formed question. For the **Noisy Background** dataset, we randomly sampled an arbitrary length phrase from the other answer and add to the original clean question. For the **Yahoo** dataset, we randomly execute those three operations to generate threefold noisy questions, which contains 254k triples. The example of ill-formed question in those datasets are shown in Table II.

| **Original well-formed question : why don't european restaurants serve water?** | |
|---|---|
| **Type** | **Ill-formed question** |
| Wrong word/typo | why don't *oeurpan rantaurest* serve *wataar*? |
| Wrong order | european restaurants why serve don't water? |
| Noisy background | concerned with the digestive process why don't european restaurants serve water? |
| Three operations | why *digistive* process with the *restarunts* european don't serve water? |

TABLE II. Example of the three operations to generate ill-formed question on Yahoo Dataset

**Customer Service Userlog (CSU)**: This anonymized dataset contains online question-answering userlog from a commercial customer service platform containing 1 million instances in chinese language. The ill-formed question is the question asked by users and the well-formed question is selected from a pool of FAQs collected by editors. After we delete the duplicated triples, the left triple size is 111k. The average length of ill-formed questions is around 6 tokens, and the average length of well-formed questions is also around 6 tokens while the average length of answers is around 54 tokens with 14k vocabulary size and 2041 characters.

### 2.4.2 Baselines and Benchmarks

The compared methods are summarized as follows:

- **Seq2Seq** is a basic encoder-decoder sequence learning system with Luong attention (Luong et al., 2015) and Bi-direction LSTM on encoder model.
- **PARA-NMT** (Dong et al., 2017) is a NMT-based question paraphrasing method which assigns higher weights to those linguistic expressions likely to yield correct answers.
- **AQA** (Buck et al., 2018) is the reinforce method seeking to reformulate questions such that the QA system has the best chance of returning the correct answer in the reading comprehension task. Since our datasets do not contain the context information, we use the QA-lstm to measure the similarity between the generated question and the answer as the reward. Following (Britz et al., 2017), we use a bidirectional LSTM as the encoder and a 4-layer stacked LSTM with attention as the decoder.

| Method | Wrong Word | | | Wrong Order | | | Noisy Background | | |
|---|---|---|---|---|---|---|---|---|---|
| | **BLEU-1** | **Rouge** | **Meteor** | **BLEU-1** | **Rouge** | **Meteor** | **BLEU-1** | **Meteor** | **Rouge** |
| Seq2Seq | 47.10 | 60.41 | 30.95 | 53.11 | 67.67 | 36.37 | 50.17 | 62.19 | 31.75 |
| PARA-NMT | 53.10 | 64.91 | 35.59 | 59.13 | 73.75 | 41.27 | 55.37 | 68.45 | 37.56 |
| TOQR | 43.15 | 56.04 | 28.47 | 49.60 | 45.49 | 56.76 | 32.75 | 57.85 | 45.39 |
| AQA | 61.93 | 77.36 | 45.91 | 63.34 | 80.83 | 50.15 | 61.08 | 74.14 | 42.10 |
| QREFINE-RF | 67.82 | 83.16 | 51.07 | 70.74 | 87.21 | 55.24 | 69.12 | 85.21 | 53.17 |
| QREFINE-PPO | **68.83** | **84.76** | **52.72** | **72.22** | **88.94** | **56.22** | **71.57** | **86.12** | **53.22** |

TABLE III. Question Generation Evaluation on Yahoo dataset to test models ability to correct wrong words, order and remove background.

- **TOQR** (Nogueira and Cho, 2017) is the query reformulation method with reinforcement learning to maximize the number of relevant documents returned. TOQR use reinforcement method to select terms from the original query and candidate retrieved document to reformulate the query, and the reward is the document recall.

Since the proposed QREFINE consists of several components, we consider several variations of QREFINE as follows:

**QR-word** is the reinforce model only using wording reward, which is viewed as word-level reward. **QR-ans** is the reinforce model using answer correlation as the reward, which is views as question-level reward. **Qrefine-RF** combines both word-level wording reward and question-level answer correlation reward and uses REINFORCE policy gradient based to optimize.

**Qrefine-PPO** is the proposed model using PPO and combining both word-level wording reward and question-level reward.

**Experimental Setting** We randomly divide the dataset into a training set (80%), a development set (10%), and a test set (10%). We tune the hyperparameters on development set and

| Method | Yahoo | | | | | | CSU | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **BLEU-1** | **BLEU-2** | **BLEU-3** | **BLEU-4** | **Rouge** | **Meteor** | **BLEU-1** | **BLEU-2** | **BLEU-3** | **BLEU-4** | **Rouge** | **Meteor** |
| Seq2Seq | 39.50 | 31.53 | 23.81 | 16.78 | 53.07 | 22.76 | 72.76 | 53.17 | 37.49 | 26.17 | 68.39 | 35.57 |
| PARA-NMT | 41.08 | 33.74 | 26.50 | 19.73 | 55.10 | 23.80 | 73.18 | 59.41 | 56.53 | 36.44 | 69.05 | 57.18 |
| AQA | 43.40 | 37.14 | 30.80 | 24.58 | 58.37 | 27.17 | 74.13 | 65.53 | 58.00 | 31.63 | 74.40 | 62.67 |
| TOQR | 31.23 | 20.69 | 12.45 | 5.89 | 41.92 | 15.19 | 48.92 | 44.69 | 40.34 | 35.73 | 65.10 | 33.00 |
| QR-word | 47.73 | 42.46 | 37.09 | 31.46 | 63.03 | 31.00 | 77.49 | 69.61 | 62.00 | 34.91 | 77.71 | 66.17 |
| QR-ans | 47.17 | 41.80 | 36.36 | 30.66 | 62.48 | 30.55 | 78.50 | 70.95 | 63.08 | 36.99 | 78.72 | 67.07 |
| QREFINE-RF | 48.72 | 44.20 | 39.58 | 34.74 | 64.60 | 32.59 | 79.71 | 72.43 | 64.59 | 37.54 | 79.96 | 68.40 |
| QREFINE-PPO | **50.90** | **47.47** | **43.91** | **40.19** | **67.41** | **35.37** | **82.55** | **75.54** | **67.63** | **40.33** | **82.74** | **71.54** |

TABLE IV. Question Generation Evaluation on Yahoo and CSU dataset.

report results on test set. We implement all model by Tensorflow using Python on a Linux server with Intel Xeon E5-2620 v4 CPU and an NVIDIA 1080Ti GPU. If the paper were accepted, we would publish code and data online.

On Yahoo dataset, we use the released skip-gram model word embedding (Mikolov et al., 2013) with 300 dimensions [1]. We fix the word representations during training. The character embedding of is 50 dimensions for each character. The number of hidden unit in character Bi-LSTM is 100, so the size of word through character embedding is 200. We add <EOS> at the end of sentences. And we set the word out of the vocabulary to <UNK>.

We choose word embedding of 200 dimensions for CSU dataset and use the Glove model (Pennington et al., 2014) to get the pre-trained word embedding. The character embedding of CSU dataset is 50 dimensions for each character. The number of hidden unit in character Bi-LSTM is 50, so the size of word through character embedding is 100.

---

[1] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

For BERT word embedding [1], it gives us 768 dimensions of the word embedding on both datasets. We combine contextual-free word embedding, BERT embedding and character embedding for each of word on both dataset.

We set the LSTM hidden unit size to 300 on CSU dataset and 500 on WikiAnswer dataset. Optimization is performed using Adam (Kingma and Ba, 2014), with an initial learning rate of 0.001; the first momentum coefficient was set to 0.9 and the second momentum coefficient to 0.999. The mini-batch size for the update is set at 64 on both datasets. During decoding, we greedily pick the generated word. Decoder stops when the <EOS> token is being generated. During reinforcement training, we set the ratio of language model reward and answer correlation reward $c_1$ as $\{0.1, 1, 10\}$, the discounted factor of RL $\lambda$ is $\{0.01, 0.1, 1\}$, the threshold $\epsilon$ is $\{0.1, 0.2, 0.3\}$ and the entropy regularization $c_2$ is $\{0.1, 1\}$. All hyperparameters of our model are tuned using the development set.

### 2.4.3 Question Generation

In this section, we give the experimental analysis to quantitatively and qualitatively evaluate the quality of generated questions.

**Quantitative Evaluation of Question Generation** To evaluate the quality of the generated well-fined question, we first use automatic metrics to quantitatively show the performance. We use the precision-based automatic metrics BLEU-1, BLEU-2, BLEU-3, BLEU-4 (Papineni

---

[1] https://github.com/hanxiao/bert-as-service

et al., 2002) which measures the average n-gram precision on a set of reference sentences, with a penalty for overly short sentences, and ROUGE (Lin, 2004) based on recall and METEOR (Banerjee and Lavie, 2005) that is based on both precision and recall to measure the generation results.

In Table III, our model performs best on each single task, wrong word, wrong order or removing noisy background. Since QREFINE uses character-level embedding, contextual-free and contextual word embedding BERT, it can better deal with the misspellings and understand the ill-formed question than all baselines. By using the word-level reward, our model can learn a better language policy, hence it can perform well on correcting the word order and wrong word. Besides, since our model considers the answer correlation as the reward, so it can capture the useful information in the question and achieves superior results on the noisy background.

For the three operations composite task, as shown in Table IV, our model performs the best on both datasets. In PARA-NMT, the paraphrased questions are very similar, and therefore there is a great chance that paraphrased questions always get the same answer. The reinforced query reformulation method TOQR reformulates the query by selecting terms from the retrieved documents by using the document recall as reward. Since the query is a list of terms, therefore, the generated sequence of terms have poor readability, resulting in lowest performance among all baselines. The original AQA refines the question for the reading comprehension. We use the QA-lstm as reward, which has similar framework as PARA-NMT but using REINFORCE optimization. Therefore AQA has the same problem as PARA-NMT. Compared to QR-ans, QR-word has a better performance, which shows that QR-word contributes more to the readability.

| Yahoo | | |
|---|---|---|
| **Dataset/Case** | **Ill-formed** | **Well-formed** |
| Yahoo/Case 1<br>Wrong word | what's the differncee between climate chnage and global wraming? | **Seq2Seq**: what's the difference between human beings and cancer cancer?<br>**TOQR**: differncee between climate chnage and global wraming oxygen weather warming<br>**AQA**: what's the difference between global and global warming war?<br>**QREFINE**: what's the difference between climate change and global warming? |
| Yahoo/Case 2<br>Wrong order | how safe to meet is it an american? | **Seq2Seq**: how to make it is an adult professional?<br>**TOQR**: how safe meet an american again powerful secretary<br>**AQA**: an animal how safe is it to meet an american?<br>**QREFINE**: how safe is it to meet an american? |
| Yahoo/Case 3<br>Noisy Background | who would able help what is string theory and how is it used? | **Seq2Seq**: what is empirical and how is it used mechanics?<br>**TOQR**: able help string theory and how is it used theory physics used model<br>**AQA**: what is turbo motion and how it is used?<br>**QREFINE**: what is string theory and how is it used? |

| CSU | | |
|---|---|---|
| **Dataset/Case** | **Ill-formed** | **Well-formed** |
| CSU/Case 1<br>Wrong word | 我想问一下为什么 优惠卷用不了 I want to ask why coubon cannot be used | **Seq2Seq**: 如何使用优惠券 How to use the coupon<br>**TOQR**: 优惠卷 优惠券 使用规则 有效期 特价商品 coubon coupon service regulations validity bargain goods<br>**AQA**: 如何获得优惠券 how can I get coupon<br>**QREFINE**: 优惠券不能使用怎么办 What should I do if the coupon cannot be used |
| CSU/Case 2<br>Wrong order | 我怎么办手机进水了 I do what phone has water damage | **Seq2Seq**: 手机丢失怎么找回 how do I find the lost phone<br>**TOQR**: 手机进水 远离水源 吸水 water damage avoid water absorb water<br>**AQA**: 手机进水了 my phone has water damage<br>**QREFINE**: 手机进水怎么办 What should I do if the phone has water damage |
| CSU/Case 3<br>Noisy Background | 忘记密码多次 登录现在没法登陆 forgot the password and repeatedly log in right now cannot log in | **Seq2Seq**: 无法登陆怎么办 What should I do when cannot log in<br>**TOQR**: 忘记密码 没法登陆 账号冻结 违规 错误密码 forget password cannot login block account number wrong password<br>**AQA**: 忘记密码怎么办 What should I do when I forget password<br>**QREFINE**: 登陆密码忘记了怎么办 What should I do when login password forgot |

TABLE V. Cases study of Generated Results on Yahoo dataset and Commercial Customer Service Userlog dataset. Typos and substitutions are shown in underscore.

QREFINE-PPO is higher than QREFINE-RF, which may because PPO method can improve the high variance problem using REINFORCE method on question refinement task.

**Case Study** To demonstrate the readability and effectiveness of the proposed method, Table V shows examples of Seq2Seq and QREFINE outputs on Yahoo and CSU dataset. As we can see, QREFINE has the ability to correct the wrong words (shown in Case 1), change the question word order for better readability (shown in Case 2) and remove the unnecessary

background in the original question (shown in Case 3). For example, "differncee" can be correct as "difference". And the refined question by our model is readable. But the question generated by other baselines cannot well express the original meaning in the ill-formed question or be misleading and also have problems like repeatedly the useless words (e.g., Seq2Seq in Yahoo/Case 1), no natural language sentence (e.g., TOQR in Yahoo/Case 1) and express the different meaning with the ill-formed question (e.g., AQA in Yahoo/Case 1). Therefore, the question generated by QREFINE is more readable than other alternatives and is able to keep the original user intention of asking the question.

### 2.4.4 Answer Retrieval

To validate the effectiveness of question refinement in helping retrieve answers in existing QA systems, we use PyLucene [1] for retrieving the answer to the search question.

**Hits@K**: The top K relevant answers retrieved by PyLucene using the search question. If the gold answer is inside of the top K retrieved answers, then the Hits@K of this search question equals 1, otherwise, it equals 0. The whole Hits@K of questions is the average development set question's Hits@K.

**Results** In Table VI, we can see the refined question by all methods can be better than the Ill-formed question, which shows that the refinement process is needed. TOQR, which aims to maximize the retrieval results achieves the good performance compared with other methods. However, our model QREFINE achieves very better performance comparing with TOQR in most

---

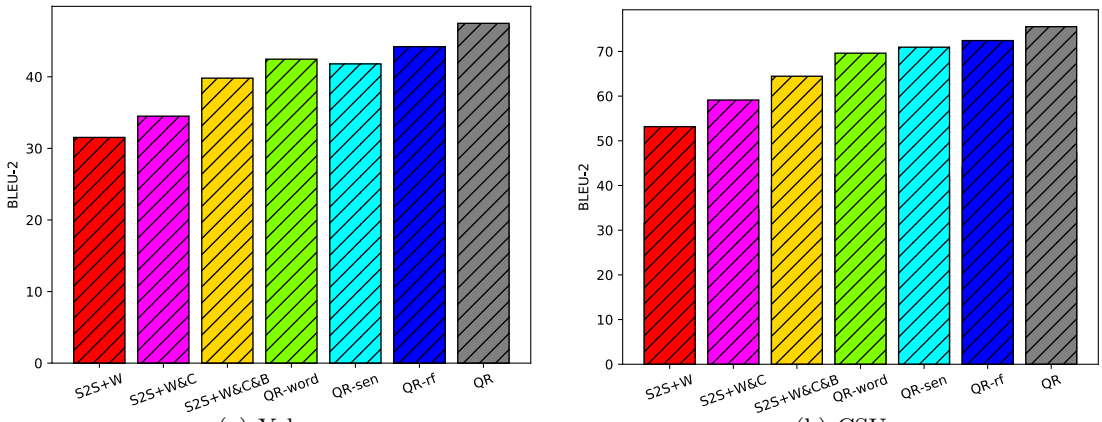
[1] http://http://lucene.apache.org/pylucene/

| **Yahoo** | **Hits@1** | **Hits@3** | **Hits@5** | **Hits@10** |
|---|---|---|---|---|
| Ill-formed | 3.35 | 4.89 | 8.68 | 14.98 |
| Seq2Seq | 3.60 | 4.50 | 8.99 | 16.97 |
| PARA-NMT | 6.47 | 7.95 | 12.01 | 17.21 |
| AQA | 6.48 | 8.12 | 13.06 | 18.78 |
| TOQR | 7.27 | 10.25 | 22.83 | **32.43** |
| QR-word | 7.11 | 9.34 | 20.79 | 29.88 |
| QR-ans | 7.16 | 9.67 | 22.45 | 30.89 |
| QREFINE | **8.09** | **10.76** | **23.95** | 32.23 |

| **CSU** | **Hits@1** | **Hits@3** | **Hits@5** | **Hits@10** |
|---|---|---|---|---|
| Ill-formed | 10.34 | 18.72 | 26.78 | 39.78 |
| Seq2Seq | 11.84 | 19.83 | 27.25 | 40.12 |
| PARA-NMT | 15.59 | 21.09 | 30.95 | 40.33 |
| AQA | 16.89 | 20.00 | 31.81 | 40.67 |
| TOQR | 20.41 | 27.98 | 34.59 | 48.78 |
| QR-word | 21.23 | 25.98 | 33.04 | 46.19 |
| QR-ans | 19.64 | 26.11 | 34.67 | 47.76 |
| QREFINE | **22.10** | **28.69** | **35.98** | **49.16** |

TABLE VI. Answering Retrieval Result on Yahoo and CSU dataset.

case and the question refined by our model has better readability. The Hits@K score retrieved by QREFINE-word is higher than Seq2Seq, which indicates that by improving the readability of question, the retrieval ability can also be improved. As QREFINE-sen performs better than Seq2Seq, it shows that the reward considering over-all question structure for a better correlation of the refined question to its answer is important. This result shows the superiority of QREFINE in greatly improving QA utility via explicitly refining the question for enhancing its readability and retrieve ability for both computer and human.



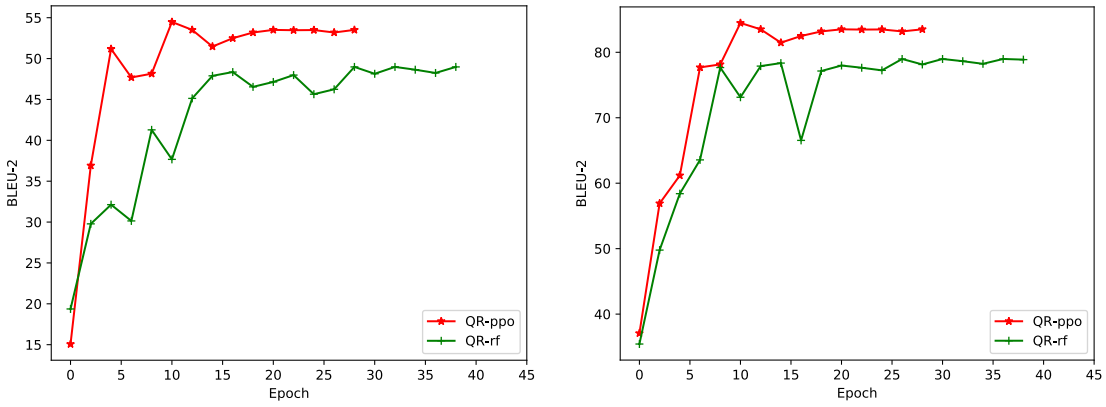

(a) Yahoo (b) CSU

Figure 2. The ablation study on Yahoo and CSU dataset

### 2.4.5 Ablation Study

In order to find out which part of the model improves the automatic evaluation performance, we do the ablation study. **S2S+W** is seq2seq model and use word-level embedding. **S2S+W&C** is seq2seq model and use word-level and char-level embedding. **S2S+W&C&B** is seq2seq model and use word-level, char-level, and BERT embedding. **QR-word** is seq2seq model considering three embeddings and using word reward to train the RL model. **QR-ans** is seq2seq model considering three embeddings and using answer coherence reward to train the model. **QR-RF** considers multi-grain word embedding and both word reward and sentence reward but uses REINFORCE method. **QR-PPO** is our model.

From 2(a) and 2(b), we can see that using multi-grain word embedding can help the model better to correct the ill-formed question than just using single word embedding. And using PPO




(a) Yahoo (b) CSU

Figure 3. The learning curve analysis on Yahoo and CSU dataset

reinforcement learning with the word-level and sentence-level reward can improve the model to learn a stable policy that can generate the appropriate well-formed question.

### 2.4.6 Learning Curves Analysis

The BLEU-2 scores and learning curves of different optimization algorithms are presented in 3(a) and 3(b). From the testing results in Table IV, we can see that the two optimization methods have comparable performance, but PPO achieves a slightly higher BLEU-2 score than REINFORCE. Moreover, we find out that the training progress of PPO is more stable than policy gradient, and the training progress of PPO is much faster. This shows that PPO methods can improve the high variance problem of using REINFORCE, and the dynamic constraint can help the learning converge quickly.

## 2.5 Related Work

### 2.5.1 Generative Text Refinement

The generative ability of deep neural networks leads to the prevalence of Seq2Seq models for reformulation task. These methods typically accomplish the reformulation by training a recurrent neural network such as an LSTM network to predict the next word in a sentence sequence. (Nogueira and Cho, 2017) uses the reinforcement learning to reformulate the query to maximize the number of relevant documents retrieved. Our work differs with their method in that we generate natural language sequences rather than terms; thus their reformulated query doesn't contain the propriety of readability and understandability. (Zhang and Lapata, 2017) proposed the sentence simplification task, which aims to make sentences easier to read and understand. Similarly, (Xie et al., 2016) operates on the character level to flexibly handle orthographic errors in spelling, capitalization, and punctuation. Although their frameworks can reform the sentences to be more readable, their objective does not include the capability of refining the question to get answers easier. Active QA (AQA) (Buck et al., 2018) uses an agent as a mediator between the user and a backbone QA system, e.g. BiDAF, to produce questions that draw out the best possible answer. Since the pretrained fixed environment, BiDAF, is not updating with the model, feedback on the quality of the question reformulations could be quite noisy which presents a challenge for training. Moreover, BiDAF works on the reading comprehension, the answer is the paraphrase in the context, therefore there is a great change that this model always generates the same answer, which brings another challenge for training.

### 2.5.2 Reinforcement Learning for QA

Due to the high dimensional action space for text generation and high diversity of the required generation result, policy gradient methods are more appropriate in the text generation than value-based methods like Q-learning (Mnih et al., 2013). By using policy gradient method, the limitation of cross-entropy loss that inherently comes with word-level optimization is alleviated and allowing sequence-level reward functions, like BLEU, to be used (Ranzato et al., 2016). (Bahdanau et al., 2017) extends this line of work using actor-critic training. There are several works that uses policy gradient on the QA task. (Liang et al., 2016) trains a semantic parser to query a knowledge base. (Seo et al., 2016) proposes query reduction networks that transform a answer-related query to search the answer in the multi-hop common sense reasoning. Li et al. (Li et al., 2017) use RL and SL to learn the paraphrase of the sentence. The on-policy method like REINFORCE suffers the high variance and slow to converge. The off-policy method like TRPO and PPO (Schulman et al., 2015; Schulman et al., 2017) recently applied on the game like Atari. They can deal with the problems by regularizing the gradient of policy. Tuan et al (Tuan et al., 2018) apply the off-policy gradient method to the sequence generation task and shows that PPO surpass policy gradient on stability and performance.

# CHAPTER 3

# ENHANCE TEXT GENERATION WITH EXTERNAL KNOWLEDGE GRAPH

This chapter was previously published as "KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning" in AAAI'21 (Liu et al., 2020). `https://www.aaai.org/AAAI21Papers/AAAI-4301.LiuY.pdf`

## 3.1 Introduction

Nowadays, numerous benchmarks for commonsense reasoning have been developed to make computers more competent and human-aware. In particular, various pre-trained approaches have achieved impressive performance on the *discriminative* commonsense tasks – i.e., AI systems are needed to select the correct option from a set of choices conditioned on a given context (Lin et al., 2020), such as CommonsenseQA (Talmor et al., 2019), COSMOSQA (Huang et al., 2019) and WinoGrande (Sakaguchi et al., 2020). However, commonsense reasoning in text generation, known as *generative* commonsense reasoning, still remains a challenge to existing models, which requires machines to generate a sentence describing a day-to-day scene using concepts from a given concept set.

In recent years, many pre-trained language generation models have been presented for text generation tasks, such as GPTs (Radford et al., 2019; Brown et al., 2020), UniLM (Dong et al., 2019), T5 (Raffel et al., 2020) and BART (Lewis et al., 2020). Although they can capture rich

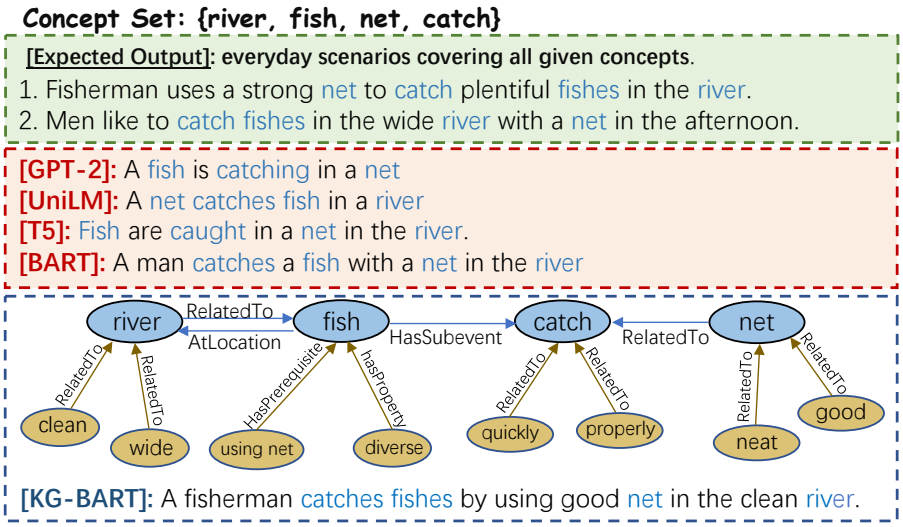



Figure 4. An example of the generation outputs of our KG-BART model (blue dotted box) and the existing models without knowledge graph augmentation (red dotted box).

language information from text sentence corpus and generate accurate language texts, almost all of them ignore knowledge information and thereby fail to generate output towards capturing the human commonsense. For example, as shown in Figure 4, given a set of commonsense concepts {*river*, *fish*, *net*, *catch*}, the task is to generate a coherent sentence describing a scenario covering all given concepts, such as "*Fisherman uses a strong net to catch plentiful fishes in the river*". From our analysis, we note that the state-of-the-art pre-trained models generate implausible and anomalous sentences in this task (red dotted box) - e.g., GPT-2 generated "*A fish is catching in a net*", UniLM generated "*A net catches fish*", etc. Moreover, the generated sentences by the pre-trained models are simple and rigid, while the human sentence is more natural and rich, like "*plentiful fishes*", "*wide river*", etc.

In this paper, we argue that only using pre-trained language models with textual concepts alone cannot provide sufficient information for generative commonsense reasoning. The commonsense knowledge graphs (KGs) (Speer et al., 2017; Sap et al., 2019a) have been developed especially for knowledge representation in symbolic systems, and they provide a lot of candidate commonsense facts mined from corpora, which have been widely used in commonsense QA tasks (Lin et al., 2019). It would be beneficial to develop a model that can exploit commonsense KGs for generative commonsense reasoning task. For example, as shown in Figure 4, by considering knowledge facts "<*fish*, *HasPrerequisite*, *using net*>" and "<*fish*, *HasSubevent*, *catch*>", it is easy to recognize the relation between concepts {*fish*, *net*, *catch*}, namely using the net to catch fish. Furthermore, the commonsense relation, like "<*river*, *RelatedTo*, *clean*>", can provide the adjunct word to facilitate generating a more natural and plausible daily scenario sentence.

In light of the fact that the knowledge graph can provide the relational information to enhance the reasoning capacity and provide adjunct words to the concept, we propose a novel Knowledge Graph-Augmented framework for generative commonsense reasoning. It has two major steps: knowledge graph grounding and graph-based encoder-decoder modeling. We first construct two KGs, one is the concept-reasoning graph and another is the concept-expanding graph, both of which encode the entity representations and their dependency relations. Secondly, we propose an encoder-decoder neural architecture, named (KG-BART), by incorporating the grounded KGs into the state-of-the-art pre-trained language generation model BART. KG-BART follows the BART architecture, but instead of using the traditional Transformer,

we introduce an effective Knowledge Graph-Augmented Transformer to capture the relations between concept set, where the grounded KGs are used as the additional inputs to the graph attention mechanism. Besides, since the token and concept entity are at different granularity levels, we integrate the text representation with the knowledge concept for relational reasoning and then disintegrate to the token-level.

Overall, the main contributions of this paper are as follows:

- To the best of our knowledge, this is the first time that the KG is incorporated into the pre-trained model to improve the ability of commonsense reasoning in text generation.
- We build the concept-reasoning graph to guide the pre-trained model to better reasoning the relationships among concepts. Moreover, we build the concept-expanding graph which considers both the inter-concept relation and intra-concept relation for KG-Augmented decoder to generate more natural and plausible output.
- We propose KG-BART, a pre-trained method that is designed to better generate language via knowledge graphs and texts, and enhance the model generalization on unseen concept sets. Particularly, the integration and disintegration components are introduced to fuse the heterogeneous information between the token and concept entity.
- The experimental results show that KG-BART significantly outperforms the state-of-the-art pre-trained models on the task of generative commonsense reasoning. Additionally, we show that KG-BART can benefit downstream tasks (e.g., commonsense QA) via generating useful context as background scenarios.

## 3.2 Problem Formulation

**Notation**. We use $\mathcal{X}$ to denote the space of all possible concept sets, and use $\mathcal{T}$ and $\mathcal{C}$ to denote the token vocabulary and concept vocabulary, respectively. The knowledge graph (KG) is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of entities, $\mathcal{E}$ is the set of edges and $\mathcal{R}$ is the set of relations among entities. For a pair of entities $v_i \in \mathcal{V}$ (subject) and $v_j \in \mathcal{V}$ (object), associated with the relation $r_{ij} \in \mathcal{R}$, the edge $e_{ij} \in \mathcal{E}$ can be represented as a triplet $(v_i, r_{ij}, v_j)$. Specifically, we assume the concept vocabulary is a subset of KG's unigram entities, namely $\mathcal{C} \subset \mathcal{V}$.

Given an unordered set of $k$ commonsense concepts $x = \{c_1, c_2, \ldots, c_k\}$, where each concept $c_i \in \mathcal{C} \subset \mathcal{X}$ is an object (noun) or action (verb), the ultimate goal of generative commonsense reasoning is to generate a natural language output $y = \{y_1, y_2, \ldots, y_l\}$ that is both correct (or valid) and natural sounding for that scenario. This is often modeled by learning a function $f : \mathcal{X} \to \mathcal{Y}$ that maps the concept set $x \in \mathcal{X}$ into a sentence $y \in \mathcal{Y}$. Our aim is to boost the performance of this task with the help of KG database $\mathcal{G}$ which can be treated as auxiliary information.

More formally, we formulate the problem as follows: $h : \{\mathcal{X}, \mathcal{G}\} \to \{\mathcal{G}^R, \mathcal{G}^E\}$ that takes the concept sets $x \in \mathcal{X}$ and the knowledge $\mathcal{G}$ as the input to first learn a concept-reasoning graph $\mathcal{G}^R$ and a hierarchical concept-expanding graph $\mathcal{G}^E$, and then $g : \{\mathcal{X}, \mathcal{G}^R, \mathcal{G}^E\} \to \mathcal{Y}$ to generate the final outputs. Specifically, $\mathcal{G}^R \subset \mathcal{G}$ consisting of all concept triplets $(v_i^R, r_{ij}^R, v_j^R)$, where $v_i^R$ and $v_j^R \in \mathcal{X}$ and $r_{ij}^R \in \mathcal{R}$ is the relation between each concept pairs. $\mathcal{G}^E = \{\mathcal{G}^R \cup \mathcal{N}(v^R)\} \subset \mathcal{G}$ is

used to enrich the graph with adjunct information, where $\mathcal{N}(v^R)$ characterizes the neighborhood relationship between concept ($v^R$) and its adjacencies in the KG database.

## 3.3 Knowledge Graph Grounding

In this section, we explain how to construct and learn the embedding representations of the concept-reasoning graph and the hierarchical concept-expanding graph from the large commonsense KG Conceptnet (Speer et al., 2017).[1]

In the generative commonsense reasoning task, traditional pre-trained methods usually encode the concept ($x$) and decode the sentence ($y$) based on text information alone, which ignore the structural information and relations between concepts and suffer from generating a lot of implausible sentences. In order to overcome this drawback, we propose to hybridize the KG and text information in the encoder and decoder modules. Specifically, in the encoder phase, we construct a concept-reasoning graph $\mathcal{G}^R$ to encompass the relations between the concept set. In the decoder phase, we construct a hierarchical concept-expanding graph $\mathcal{G}^E$ to enrich the concept structure with the neighborhood correlation preserved in the KG database. Based on our assumption, each concept corresponds to a KG's unigram entity, so we can directly match the concept set to the entities from KG to generate $\mathcal{G}^R$. In order to establish $\mathcal{G}^E$, we couple $\mathcal{G}^R$ with the association of selected neighboring nodes with each concept in KG. For many concepts, there are hundreds or thousands of neighboring nodes connected with each of them (via triplets) in KG, which provide us not only rich information but also less important or less

[1]https://github.com/commonsense/conceptnet5/wiki/Downloads

relevant entities that may be undesirable. For instance, given a concept-set {*ski*, *skier*, *mountain*}, considering the adjunct concepts for "*mountain*", "*snowy*" is more precise than others like "*small*" or "*flat*" according to the close semantics of "*snowy*" and "*ski/skier*". Based on this fact, we rank the neighboring nodes of each concept according to the word similarity scores and select their potential top-$k$ neighboring nodes adding to $\mathcal{G}^R$, so as to get $\mathcal{G}^E$. To calculate the word similarity scores, we use the pre-trained GloVe embedding (Pennington et al., 2014) as the representation of each entity node in KG. The ranking score for a particular neighboring node is the sum of similarity scores with all concepts. Here we use the cosine similarity for its simplicity and wide application.

Since some of concept pairs do not have a direct connection in the KG and some of the concept pairs connect by multiple relations, instead of directly using $\mathcal{G}^R$ and $\mathcal{G}^E$, we use a knowledge embedding method named TransE (Bordes et al., 2013) to learn their entity and relation embeddings. To prepare the training triplets of TransE model, we first collect the triplets in the one-hop path, two-hop path, and three-hop path between each concept pair. Moreover, we further collect the triples between each concept node and their neighboring nodes as follows: if the concept node is the object (noun), only the neighboring node containing the adjective word will be selected; if the concept node is action (verb), only the node containing adverb word will be selected. TransE model is trained based on those selected triplets, which generates the node embedding $\mathbf{v}_i \in \mathbb{R}^{d_e}$ for each node $v_i$ and relation embedding $\mathbf{r}_{ij} \in \mathbb{R}^{d_r}$ for each edge $e_{ij}$. For $\mathcal{G}^R$, we denote each concept embedding as $\mathbf{v}^R$, and relation embeddings as $\mathbf{r}^R_{ij} = \mathbf{v}^R_i - \mathbf{v}^R_j$ instead of the output of TransE to avoid missing relations between concepts.

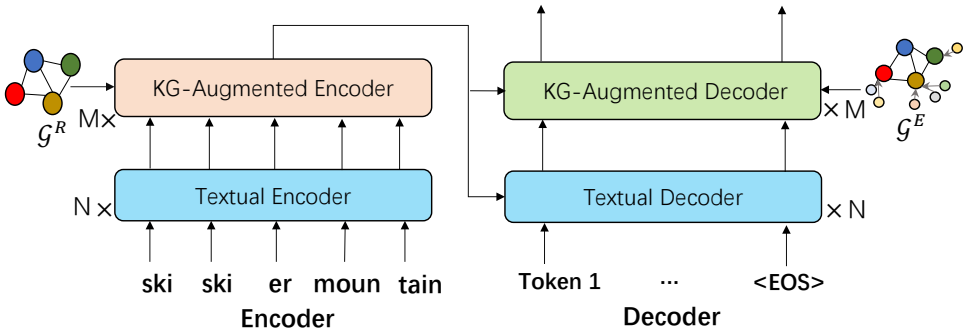



Figure 5. The proposed KG-BART model.

For $\mathcal{G}^E$, since those neighboring nodes are connected with the concepts in the KG, we directly add their node embeddings $\mathbf{v}^N$ and relation embeddings $\mathbf{r}^N$ to $\mathcal{G}^R$.

### 3.4 Graph-Based Encoder-Decoder Modeling

**Overview**. Figure 5 presents an overview of the proposed KG-BART model, which follows the BART encoder-decoder architecture but uses both text concepts and KG as the input. The encoder is composed of two components: one traditional textual Transformer encoder module (Vaswani et al., 2017) to represent the contextual information of each token; and another KG-augmented Transformer module based on graph attention mechanism to integrate the entity-oriented knowledge information into token representation. Similarly, the decoder is also composed of a stack of a textual Transformer decoder module and a KG-augmented Transformer decoder module to generate sentences with the ability of commonsense reasoning. Specially, we use a hierarchical graph attention mechanism to refine the KG-augmented decoder to capture the inherent structural correlations of intra-concept and inter-concept in the graph.

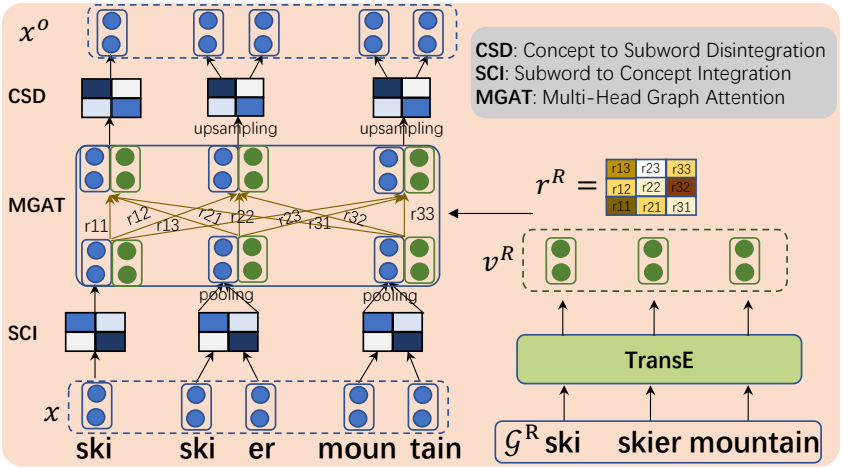



Figure 6. The KG-augmented encoder.

Note that all the node and relation embeddings are held fixed in the training process of KG-BART. Since our textual Transformers are the same as that used in BART, here we exclude a comprehensive description of these modules and refer readers to (Lewis et al., 2020) and (Vaswani et al., 2017) for more details. In the following, we will focus on the proposed KG-augmented Transformer.

### 3.4.1 KG-Augmented Encoder

As shown in Figure 6, above the textual encoders, the KG-augmented encoder is designed to enrich the token representation by considering the KG structure. We propose to incorporate graph representations into the neural encoding process via a graph-informed attention mechanism. It takes advantage of the explicit relations to learn better intra-concept relations. Formally, the KG-augmented encoder integrates the input token embeddings $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, which is

the output of the textual encoders, as well as the embedding of concept-reasoning graph $\mathcal{G}^R$ to update the token representation as $\{\mathbf{x}_1^o, \ldots, \mathbf{x}_n^o\}$. **Subword to Concept Integration (SCI)** As the input token embeddings are based on a sequence of subwords, while our concepts in the KG are at word-level, we need to align these different granularity sequences. To apply the relation between concepts, we group the subwords for each concept. In particular, we adopt one convolutional neural network (CNN) (Kim, 2014) with a max-pooling layer to efficiently obtain the representation in word-level.

Here we take a concrete concept as an example to better illustrate this process. Supposing that a concept $c_i$ is made up of a sequence of subwords $\{x_1, x_2, \ldots, x_m\}$, where $m$ is the number of subwords. Given the token embeddings $\mathbf{x}$ from textual encoder, we first utilize a Conv1D layer, $\mathbf{x}'_t = \mathbf{Z}\left(\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+l-1}\right)^T, t \in [1, m-l+1]$, where $\mathbf{Z} = [z_1, \ldots, z_l] \in \mathbb{R}^{1\times l}$ is trainable parameters and $k$ is the kernel size. We then apply a max-pooling layer over a sequence of the output embeddings after Conv1D:

$$\mathbf{e}\left(c_i\right) = \text{MaxPooling}\left(\mathbf{x}'_1, \ldots, \mathbf{x}'_{m-l+1}\right). \tag{3.1}$$

Therefore, the final word-level textual embedding of concept is represented as $\mathbf{e}^w = \{\mathbf{e}(c_1), \ldots, \mathbf{e}(c_l)\} \in \mathbb{R}^{k\times d_w}$ where $d_w$ denotes the dimension of concept embedding.

**Multi-Head Graph Attention (MGAT)** Given the embedding representation of concept-reasoning graph $\mathcal{G}^R$ with node features $\mathbf{v}^R \in \mathbb{R}^{k\times d_e}$ and relation features $\mathbf{r}^R$, we apply the graph attention networks (GAT) (Veličković et al., 2017) to iteratively update the representations for

each concept $\mathbf{v}_i^R$ through its neighbors $\mathcal{N}_i^R$. We denote the word-level hidden state as $\mathbf{h}_i \in \mathbb{R}^{d_h}$, where $i \in (1, \ldots, k)$. We further modify the GAT layer to infuse the pairwise relation embedding $\mathbf{r}_{ij}^R \in \mathbb{R}^{d_r}$. Therefore, the multi-head graph attention can be denoted as:

$$\begin{aligned}
&\mathbf{H} = [\mathbf{e}^w;\ \mathbf{W}_e\mathbf{v}^R], \\
&z_{ij} = \text{LeakyReLU}\left(\mathbf{W}_a\left[\mathbf{W}_q\mathbf{h}_i; \mathbf{W}_k\mathbf{h}_j; \mathbf{W}_r\mathbf{r}_{ij}^R\right]\right), \\
&\alpha_{ij} = \frac{\exp\left(z_{ij}\right)}{\sum_{l=1}^{|\mathcal{N}_i^R|}\exp\left(z_{il}\right)}, \quad \mathbf{h}_i' = \|_{k=1}^{K}\sigma\left(\sum_{j=1}^{|\mathcal{N}_i^R|}\alpha_{ij}^k\mathbf{W}_v^k\mathbf{h}_i\right),
\end{aligned} \tag{3.2}$$

where $K$ is the multi-head number, $\|_{k=1}^{K}$ denotes an operation of multi-head used in Transformer, which concatenates the attention embeddings from different heads and feeds the result into a linear projection. $\mathbf{W}_a, \mathbf{W}_e, \mathbf{W}_r, \mathbf{W}_q, \mathbf{W}_k$ and $\mathbf{W}_v$ are trainable weights and $\alpha_{ij}$ is the attention weight between $\mathbf{h}_i$ and $\mathbf{h}_j$. The word-level hidden state $\mathbf{H}$ contains the latent dependencies between any two concepts from textual aspect information $\mathbf{e}^w$ and KG aspect information $\mathbf{v}^R$. And $\mathbf{r}^R$ incorporates relation representations as prior constraints into the encoding process. In this way, our model can learn better and richer concept representations containing the relationship among concepts.

**Concept to Subword Disintegration (CSD)** After updating the word-level hidden state considering the relation between concepts in the KG, we need to disintegrate the concept to the subword-level for the following process. We first upsample word-level hidden state $\mathbf{h}_i'$ with $(m - l + 1)$ times (the length before MaxPooling) as $[\mathbf{h}_i'^1, \ldots, \mathbf{h}_i'^{m-l+1}]$ and utilize a Deconv1D

layer with vector $\mathbf{Z} = [z_0, \ldots, z_l] \in \mathbb{R}^{1\times l}$ used in Conv1D to form the Deconv1D matrix $\mathbf{Z}_D \in \mathbb{R}^{m\times(m-l+1)}$ to get the subword-level hidden state $\mathbf{u}_i$:

$$[\mathbf{u}_i^1, \ldots, \mathbf{u}_i^m]^T = \begin{pmatrix} z_0 & & & \\ \cdots & z_0 & & \\ z_l & \cdots & \cdots & \\ & z_l & & z_0 \\ & & & \cdots \\ & & & z_l \end{pmatrix} * \begin{pmatrix} \mathbf{h}_i'^1 \\ \mathbf{h}_i'^2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{h}_i'^{m-l+1} \end{pmatrix}. \tag{3.3}$$

Then, a two-layer feed-forward network with GeLU activation (Hendrycks and Gimpel, 2016b) function and a residual layer normalization are applied to obtain the final output can be represented $\mathbf{x}_i^o$:

$$\begin{aligned} \mathbf{p}_i &= \mathbf{W}_{o2}\,\mathrm{GeLU}\left(\mathbf{W}_{o1}\left(\mathbf{u}_i + \mathbf{x}_i\right)\right), \\ \mathbf{x}_i^o &= \mathrm{LayerNorm}\left(\mathbf{p}_i + \mathbf{x}_i\right), \end{aligned} \tag{3.4}$$

where $\mathbf{W}_{o1} \in \mathbb{R}^{d_f\times d_h}$ and $\mathbf{W}_{o2} \in \mathbb{R}^{d_h\times d_f}$ are learnable parameters, $d_f$ is the hidden size of the feedforward layer.

### 3.4.2 KG-Augmented Decoder

In this section, our KG-augmented decoder, as shown in Figure 7, incorporates hierarchical graph structure into the decoding process to capture the relations between concepts and their neighboring nodes which can help to generate more precise and natural output. To embody the hierarchical concept-expanding graph $\mathcal{G}^E$ with the generation process, we propose the multi-head hierarchical graph attention layer.

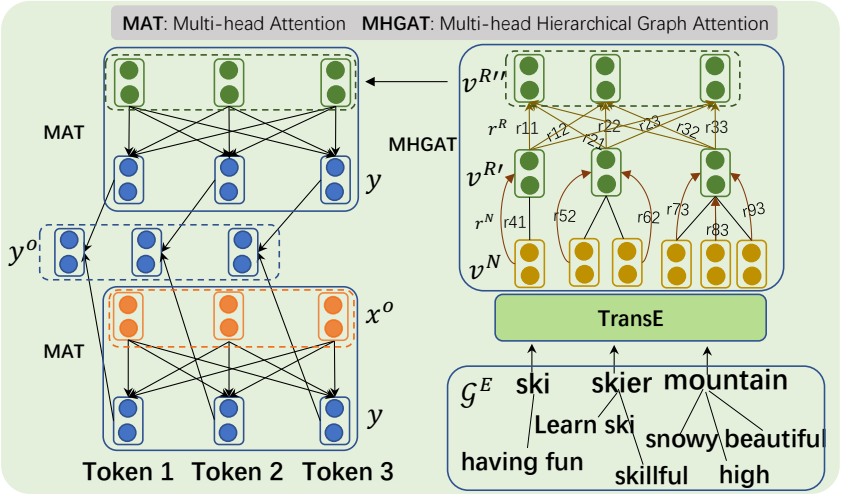



Figure 7. The KG-augmented decoder.

**Multi-Head Hierarchical Graph Attention (MHGAT)** To contain the adjunct description for the concept node, the first layer of hierarchical graph attention is to update the concept node $\mathbf{v}_i^R \in \mathbb{R}^{de}$ through its inter-concept neighboring nodes $\mathcal{N}_i^N$ with relation embedding $\mathbf{r}_{ij}^N \in \mathbb{R}^{dr}$.

$$
\begin{aligned}
z_{ij} &= \text{LeakyReLU}\left(\mathbf{W}_a\left[\mathbf{W}_q\mathbf{v}_i^R; \mathbf{W}_k\mathbf{v}_j^N; \mathbf{W}_r\mathbf{r}_{ij}^N\right]\right), \\
\alpha_{ij} &= \frac{\exp\left(z_{ij}\right)}{\sum_{l=1}^{|\mathcal{N}_i^N|}\exp\left(z_{il}\right)}, \quad \mathbf{v}_i^{R\prime} = \Big\|_{k=1}^{K}\sigma\left(\sum_{j=1}^{|\mathcal{N}_i^N|}\alpha_{ij}^k\mathbf{W}_v^k\mathbf{v}_j^R\right).
\end{aligned}
\tag{3.5}
$$

After updating the concepts with their neighboring nodes, the concepts get their new embedding $\mathbf{v}^{R\prime}$. The second graph attention layer updates the concept representation considering the intra-concept relations $\mathbf{r}_{ij}^{R} \in \mathbb{R}^{dr}$.

$$
\begin{aligned}
z_{ij} &= \text{LeakyReLU}\left(\mathbf{W}_a\left[\mathbf{W}_q\mathbf{v}_i^{R\prime}; \mathbf{W}_k\mathbf{v}_j^{R\prime}; \mathbf{W}_r\mathbf{r}_{ij}^{R}\right]\right), \\
\alpha_{ij} &= \frac{\exp\left(z_{ij}\right)}{\sum_{l=1}^{|\mathcal{N}_i^R|}\exp\left(z_{il}\right)}, \quad \mathbf{v}_i^{R\prime\prime} = \|_{k=1}^{K}\sigma\left(\sum_{j=1}^{|\mathcal{N}_i^R|}\alpha_{ij}^{k}\mathbf{W}_v^{k}\mathbf{v}_j^{R\prime}\right).
\end{aligned} \tag{3.6}
$$

We further compute the two multi-head attention (MAT) (Vaswani et al., 2017) to capture textual and KG influence. One is the attention between the encoder hidden state $\mathbf{x}^o$ and the previously generated token hidden state $\mathbf{y}$. The other is the attention between the updated concept embeddings $\mathbf{v}^{R\prime\prime}$ and the previously generated token hidden state $\mathbf{y}$ as follows:

$$
\text{AT}^{\text{KG}} = \text{MAT}(\mathbf{y}, \mathbf{v}^{R\prime\prime}, \mathbf{v}^{R\prime\prime}), \quad \text{AT}^{\text{TX}} = \text{MAT}(\mathbf{y}, \mathbf{x}^o, \mathbf{x}^o).
$$

The final decoder output is the concatenate of the two attention with a residual connection as:

$$
\mathbf{y}^o = \mathbf{W}_{att}[\text{AT}^{\text{KG}};\ \text{AT}^{\text{TX}}] + \mathbf{y}, \tag{3.7}
$$

where $\mathbf{W}_{att} \in \mathbb{R}^{d_h \times 2d_h}$ is the trainable weight. $\mathbf{y}^o$ is used to predict the token sequence: $P_{\text{vocab}} = \text{softmax}\left(\mathbf{W}_{\text{out}}\mathbf{y}^o + \text{b}_{\text{out}}\right)$, $\mathbf{W}_{att} \in \mathbb{R}^{V \times d_h}$ and $V$ is the vocabulary size.

### 3.4.3 KG-BART Model Pre-Training

The embedding vectors of words in text and nodes/entities in KG are obtained in separate ways, making their vector-space inconsistent. In order to fuse the KG into BART, similar to BART, KG-BART is trained by corrupting texts and then optimizing a reconstruction loss,

the cross-entropy, between the decoder's output and the original texts. We randomly select five concept nodes from our selected entities and mask some concepts among them. KG-BART still takes the entity and relation embedding of all concepts without considering whether the token is masked. Since the graph in the decoder only contains the concept set entities, the decoder is modified as without updating the concept nodes with their neighboring nodes in the pre-training stage. KG-BART is pre-trained to generate the original concept token from the masked concept nodes. For example, "*[mask] wound [mask] teach soldier*" in the encoder and "*student wound treat teach soldier*" in the decoder. The number of the masked token is randomly sampled from 0 to 5.

## 3.5 Experiment and Analysis

### 3.5.1 Dataset

CommonGen (Lin et al., 2020) is a constrained text generation task, which is to explicitly test the ability of machines on commonsense reasoning when generating a text. The dataset declared in this task is constructed through a combination of crowdsourced and existing caption corpora. Statistically speaking, it consists of 77k commonsense descriptions over 35k unique concept sets. In average, each concept set is composed of 3∼5 unique concepts. We present the basic statistics of this dataset in Table VII. It's worth to notice that all pairs of concepts in the test concept set are unseen in the training data so that it poses a challenge for text generalization.

**Training Details and Parameters** To implement the TranE model for KG embedding, we use the open source OpenKE,[1] and dimension of entity embedding $d_e$ and relation embedding $d_r$ to 1,024. The quantity of the select concepts for training TransE is 12K and covers all concept entities in CommonGen. In the pre-training procedure of KG-BART, we sample 200K five-concept sets from those select concepts. The entity embeddings and relation embeddings are fixed during pre-training. Since the pre-training is computation costly, we start pre-training from BART's released checkpoint and randomly initialize KG-Augmented Transformer in KG-BART with $\mathcal{N}(0, 0.02)$. We further train KG-BART for 0.2 million steps on a Nvidia Titan-RTX 24GB GPUs.

Our implementation of KG-BART is based on BART code,[2] which is implemented based on PyTorch. In detail, we have the following model size: the layer number of Textual Transformer $N = 6$, the layer number of KG-Augmented Transformer $M = 6$, the dimension of token embedding $d_w = 1024$, multi-heads $K = 16$ and the kernel size $l$ of CNN is set to 2. We tokenize the text using the byte-pair encoding same as GPT-2 (Radford et al., 2019), with the maximum length of 32 for encoder and 64 for decoder. We used AdamW (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e-6$ for optimization. We set the initial learning rate from $\{8e-6, 1e-5, 2e-5, 3e-5\}$ with warm-up rate of 0.1 and $L_2$ weight decay of 0.01. The batch size is selected from {16, 24, 32}. We employ half-precision training (floating

---

[1]https://github.com/thunlp/OpenKE

[2]https://github.com/huggingface/transformers

points 16) using `apex`[1] to reduce memory consumption and speed-up training. All models is trained with maximum likelihood estimation with the label smoothing and smoothing factor 0.1 (Szegedy et al., 2016). In the fine-tuning process, the model is trained with a maximum number of 5 epochs and the gradients are accumulated every four steps. We apply dropout with probability 0.1 to avoid over-fitting. During inference, we use beam search with beam size being 5 and length penalty with factor being 0.6.

### 3.5.2 Baselines

We compare the performance of our proposed model with several state-of-the-art pre-trained text generation models. **GPT-2** (Radford et al., 2019) is an unidirectional model to predict tokens given the input text in an auto-regressive manner. **UniLM** (Dong et al., 2019) proposes a unified model of language understanding and language generation using the masked language modeling. **UniLM2** (Bao et al., 2020) further proposes a pseudo-masked language model to learn intra-relations between masked spans via partially auto-regressive modeling. **BERT-Gen** (Bao et al., 2020) fine-tunes BERT for sequence-to-sequence language generation using a similar training objective employed by UniLM. **T5** (Raffel et al., 2020) introduces a unified framework that processes all text-based language problems into a text-to-text generation format. **BART** (Lewis et al., 2020) introduces a denoising autoencoder for pre-training sequence-to-sequence models. For the implementation of those models for the generative commonsense reasoning task, we refer readers to (Lin et al., 2020) for more details.

---

[1]https://github.com/NVIDIA/apex

| | Train | Dev | Test |
|---|---|---|---|
| # **Concept sets** | 32,651 | 993 | 1,497 |
| # **Sentences** | 67,389 | 4,018 | 6,042 |
| % **Unseen Concepts** | - | 6.53% | 8.97% |
| % **Unseen Concept-Paris** | - | 96.31% | 100.00% |
| % **Unseen Concept-Triples** | - | 99.60% | 100.00% |

TABLE VII. The basic statistics of the CommonGen dataset.

| **Model\Metrics** | **BLEU-3/4** | | **ROUGE-2/L** | | **METEOR** | **CIDEr** | **SPICE** | **Coverage** |
|---|---|---|---|---|---|---|---|---|
| **GPT-2** (Radford et al., 2019) | 30.70 | 21.10 | 17.18 | 39.28 | 26.20 | 12.15 | 25.90 | 79.09 |
| **BERT-Gen** (Bao et al., 2020) | 30.40 | 21.10 | 18.05 | 40.49 | 27.30 | 12.49 | 27.30 | 86.06 |
| **UniLM** (Dong et al., 2019) | 38.30 | 27.70 | 21.48 | 43.87 | 29.70 | 14.85 | 30.20 | 89.19 |
| **UniLM-v2** (Bao et al., 2020) | 31.30 | 22.10 | 18.24 | 40.62 | 28.10 | 13.10 | 28.10 | 89.13 |
| **T5-Base** (Raffel et al., 2020) | 26.00 | 16.40 | 14.57 | 34.55 | 23.00 | 9.16 | 22.00 | 76.67 |
| **T5-Large** (Raffel et al., 2020) | 39.00 | 28.60 | 22.01 | 42.97 | 30.10 | 14.96 | 31.60 | 95.29 |
| **BART** (Lewis et al., 2020) | 36.30 | 26.30 | 22.23 | 41.98 | 30.90 | 13.92 | 30.60 | 97.35 |
| **Human Performance** | 48.20 | 44.90 | 48.88 | 63.79 | 36.20 | 43.53 | 63.50 | 99.31 |
| **KG-BART** | **42.10** | **30.90** | **23.38** | **44.54** | **32.40** | **16.83** | **32.70** | **98.68** |

TABLE VIII. Experimental results of different baseline methods on the CommonGen test dataset. We show the best results in boldface, and those with the second best performance are underlined.

### 3.5.3 Automatic Evaluation

Following the previous generation tasks, we use several widely-used automatic metrics to evaluate the performance, such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004), which mainly focus on measuring n-gram similarities. We provide the Coverage of concept, which is average percentage of the input concepts that are present after lemmatization. Moreover, we apply evaluation metrics specially used on image captioning task, such as SPICE (Anderson et al., 2016) and CIDEr (Vedantam et al., 2015).

These metrics focus on evaluating the associations between mentioned concepts instead of n-gram overlap. To evaluate the human performance within each metric, we treat each ground-truth sentence in test dataset as a *system prediction* and compare it with other references.

Table VIII presents the experimental results in a variety of metrics and methods reported on the Leaderboard.[1] We can see that KG-BART performs best among all the pre-trained models. KG-BART outperforms 7.95%/ 8.04% on BLEU-3/4 than the second best model T5-large. KG-BART gains 1.15 improvements than the second best model BART on ROUGE-2, the gain 0.67 than UniLM on ROUGE-L. KG-BART gains 1.50 on METEOR than the second best model BART. KG-BART beats the second best model T5-large by 12.50% on CIDEr and 3.48% on SPICE. Moreover, KG-BART gets the highest Coverage 98.68 among all baseline pre-trained models. The results suggest that leveraging the pre-trained generation model with the knowledge graph can improve the performance of generative commonsense reasoning.

### 3.5.4 Human Evaluation

The automatic evaluations are unable to measure the coherence of the generated text properly. Therefore, we also access system performance by human evaluation. We randomly select 100 instances on the CommonGen development set and invite 3 annotators to independently score the outputs of different models. Annotators need to give overall quality of generative commonsense sentence by scoring them from 1 (worst) to 5 (best) taking into account the following four criteria: (1) Rationality: is the sentence the reasonable commonsense scenario? (2)

---

[1]https://inklab.usc.edu/CommonGen/leaderboard.html

| Model | 1 | 2 | 3 | 4 | 5 | Rating |
|---|---|---|---|---|---|---|
| **GPT-2** | 22% | 16% | 23% | 20% | 19% | 2.98 |
| **UniLM** | 5% | 17% | 22% | 24% | 32% | 3.61 |
| **T5-large** | 2% | 15% | 12% | 32% | 39% | 3.91 |
| **BART** | 1% | 10% | 17% | 30% | 42% | 4.02 |
| **KG-BART** | 0 % | 8% | 12% | 25% | 55% | **4.27** |

TABLE IX. Ranking results of system outputs using human evaluation.

Fluency: is the sentence fluent and grammatical? (3) Succinctness: does the sentence avoid repeating information? (4) Naturalness: does the sentence use adjunct words? The larger rating denotes a better summary quality. The rating score of each system is computed by averaging the performance scores of all test instances.

Table IX shows the final results of five methods, where both the percentage of ranking and overall ratings are reported. The results demonstrate that KG-BART is able to generate higher quality output than other models. Specifically, the outputs generated by KG-BART generally possesses more reasonable scenario and are more coherent and precise comparing with the other models. The human evaluation results additionally validate the effectiveness of the proposed model. Moreover, based on the 100 final scores for each approach, we conduct Wilcoxon signed-rank tests (Wilcoxon et al., 1970). Comparing KG-BART with T5-Large and BART, the $p$-values of Wilcoxon signed-rank testing at 95% confidence level are $1.2e{-}4$ and $2.9e{-}3$, which mean the improvements achieved by our approach are statistically significant.

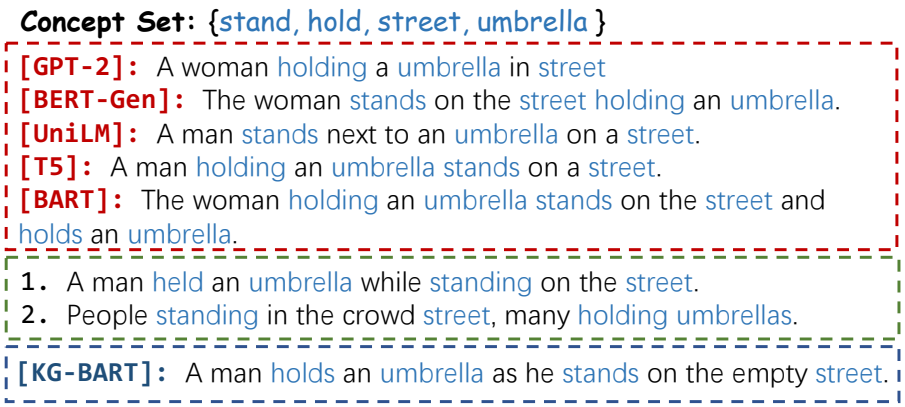



Figure 8. A case study of a specific concept set {*stand*, *hold*, *street*, *umbrella*} for qualitative analysis of machine generations. Human references are collected from AMT.

### 3.5.5 Case Study

Figure 8 gives a specific input concept set {*stand, hold, street, umbrella*}, together with the text generations of different models and human references. We find that the outputs of fine-tuned pre-trained language models have several problems: (1) not covering all concepts, e.g., GPT-2 only covers "*hold*, *umbrella*, *street*", ignoring the "*stand*", (2) unreasonable commonsense relationship between concepts, e.g. in UniLM, the output "*A man stands next to an umbrella on a street*" is a rare scenario in daily life, and (3) repeating the same content and incorrect grammar, e.g. in BART, it uses both "*holding an umbrella*" and "*holds an umbrella*", which is repeated information, and in GPT-2, the indefinite article of "*umbrella*" should be "*an*" rather than "*a*". By contrast, the output generated by KG-BART covers all concepts and is a relatively reasonable scenario and is comparatively as natural and plausible as the references stated by human.

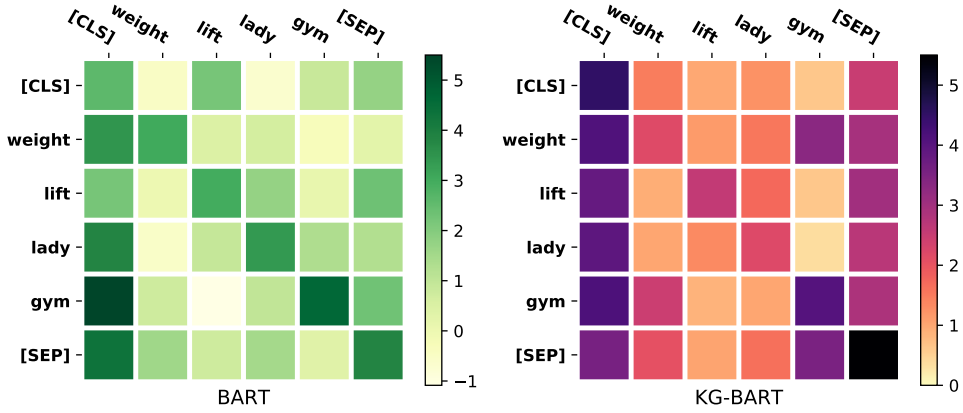



Figure 9. Attention weights of the last layers of BART and KG-BART encoder.

We also visualize the attention weights of the last layers of KG-BART and BART encoder to validate that our model can capture the better relationship between concepts, as shown in Figure 9. We can see that the related concept pairs in KG-BART attend much more attention, which is consistent with that in the knowledge graph. For example, in practice, "*weight*" has a strong relationship with "*gym*" on the knowledge graph and the attention weight between them should be large. However, this strong relationship has not been demonstrated in BART without knowledge graph. Therefore, it is reasonable to introduce a knowledge graph as relationship augmentation for better concept representation, also as a guidance to generate more reasonable sentences further.

### 3.5.6 Error Analysis

We investigate the error cases found by examining the generated sentences with low evaluation scores and find three types of errors:

The first error is that our KG-BART tends to generate a long sentence to cover the concept set. For example, given a concept set "{*talk phone wear*}", our KG-BART will generate "*A man and a woman are talking on the phone and one of them is wearing glasses.*", while the human ground truth is "*A man wearing glasses is talking on a phone.*"

The second error of our model is that KG-BART suffers repeatedly generating the same concept. For example, given the concept set "{*roll ball lane pin*}", our KG-BART generates "*A man rolls a bowling ball down a bowling lane and pins the ball down the lane.*", while the human ground truth is "*The bowling ball rolled straight down the center of the lane and knocked down all of the pins.*"

The third error of our model is that in some sentences, the generated sentences by KG-BART are still different from human commonsense. For example, given a concept set "{*jump water cliff watch*}", our KG-BART will generate "*The boy jumped off the cliff to watch the water.* ", while the human ground truth is "*Watch him jumping from the cliff to the water.*"

We think those limitation caused by KG-BART can only learn well the local relation between each concept pair, which means learning the pattern between concept pairs, so the proposed method is good at generating correct phrase. But it fail to capture the global relationship which need to check the relation between phrase in the generated sentence. How to capture this global relationship will be our future work direction.

### 3.5.7 Ablation Study

To evaluate the contributions of individual components of our proposed framework, we conduct ablation analysis to investigate the following research questions: (1) whether the KG-

| Ablation methods | | BLEU-3/4 | ROUGE-2/L |
|---|---|---|---|
| (1) KG-Aug Enc. ✓ | Dec. ✗ | 40.40/29.40 | 22.66/43.13 |
| (2) SCI ✗ | CSD ✗ | 41.20/29.70 | 23.15/43.57 |
| (3) MGAT ✗ | MHGAT ✗ | 40.90/29.30 | 22.96/43.78 |
| (4) Pre-training✗ | | 39.80/27.90 | 21.87/42.92 |

TABLE X. Ablation study of the proposed model. SCI, CSD, MGAT and MHGAT are KG-BART components.

augmented encoder and decoder improves the performance? (2) whether KG-BART is good at incorporating entity embedding with Transformer? (3) does the KG-BART pre-training works?

To this end, we test on the following ablations: (1) textual Transformer with only KG-augmented encoder; (2) using the same entity representation at each subword position rather than using SCI and CSD; (3) concatenate the entity embedding with word embedding rather than using MGAT and MHGAT; and (4) without the KG-BART pre-training. Table X summarizes the ablation results. It shows that KG-BART can still outperform all these four variants, certifying the effectiveness of each designed component in our model and we can also see that incorporating KG with the pre-trained model can help the model achieve a better performance.

### 3.5.8 Transfer KG-BART to Commonsense QA

We also investigate whether the ability of generative commonsense reasoning in KG-BART can benefit commonsense-centric downstream tasks such as Commonsense Question Answering (CSQA) (Talmor et al., 2019). We use the models trained on the CommonGen dataset for generating useful context to the question. We draw out the *nouns* and *verbs* in questions and

five choices. We combine the concepts of question $q$ with each choice $c_i$ to build concept sets. Then, we construct the concept-reasoning and concept-expanding graphs based on concepts and use these concept sets and the graphs as inputs to KG-BART to generate the context sentence $g_i$ for each choice. Finally, we prepend the outputs in front of questions, i.e., "*<s>G:*$g_i$ *</s> Q:*$q$ *</s> C:*$c_i$ *</s>*". The RoBERTa (Liu et al., 2019a) model uses the same form without "*G:*$g_i$ *</s>*" in fine-tuning stage for CSQA. We present the learning curve in Figure 10. In detail, $X$ axis is the number of training steps and $Y$ axis is the accuracy on official dev dataset.

We find that in most cases, using the context generated by pre-trained models can further improve the performance of original RoBERTa by a large margin. Especially, KG-BART converges at better accuracy from 76.22 (in original RoBERTa) to 79.31 and it outperforms other baselines. We find that the context generated by our model KG-BART can speed up training about 2.5 times, if we look at the 550th steps of KG-BART (75.51) and 1,400th steps of original RoBERTa (75.31). Note that in the beginning training steps, GPT-2 causes negative transfer due to the low quality of generated context. Through manual analysis, we find that KG-BART generate more rational and natural sentences with the correct choice while generate more noisy sentences with the wrong choices. For instance, $q$="*What would you do if you want to be able to earn money?*", $c_i$="*apply for job*" (correct) with $g_i$="*applying for a job so i would earn money.*"; $c_j$="*stand in line*" (wrong) $g_j$="*i would want to earn money standing in line to get a deal on a product.*"

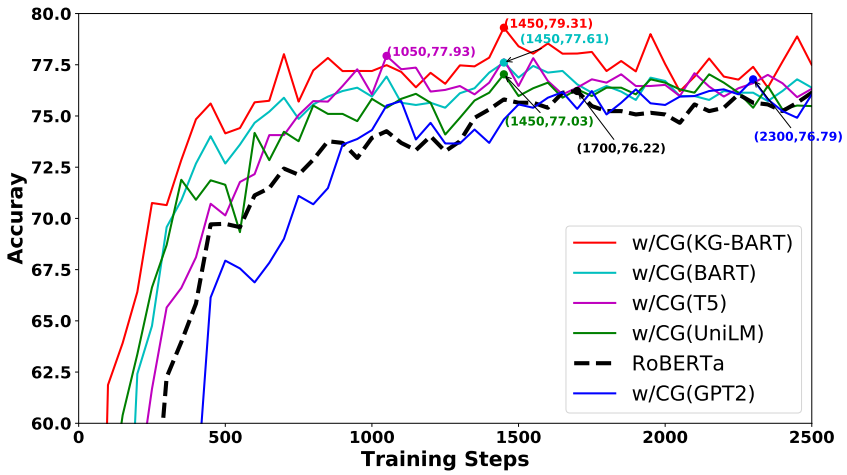



Figure 10. The learning curve of transfer study on CSQA.

## 3.6 Related Work

### 3.6.1 Enhancing NLG with Commonsense

Recently, there are a few works that enhance commonsense knowledge with language generation tasks for example storytelling (Guan et al., 2019), visual storytelling (Yang et al., 2019b), essay generation (Yang et al., 2019a), image captioning (Lu et al., 2018), evidence generation (Liu et al., 2020) and conversational generation systems (Zhang et al., 2020a). The great performance of those works suggest that generative commonsense reasoning has considerable potential to benefit downstream applications. To the best of our knowledge, the proposed model KG-BART is the first work on equipping the pre-trained language generation model with the external commonsense knowledge for the constrained language generation.

### 3.6.2 Enhancing Pre-Trained Model with Knowledge

Recently, several works have attempted to learn joint representation learning of words and entities for effectively leveraging external KGs on language understanding tasks and achieved promising results. ERNIE (Zhang et al., 2019b) incorporates informative entities from KG aligning with context to enhance pre-training language understanding. KEPLER (Wang et al., 2020b) encodes textual descriptions of entities with a pre-trained language understanding model, and then jointly optimize the knowledge embedding and language modeling objectives. K-BERT (Liu et al., 2020) injects domain knowledge into the models by adding triples from the knowledge graph as supplementary words. Inspired by these works, we argue that extra knowledge information can effectively benefit existing pre-training models on the language understanding tasks. In this paper, we utilize KGs to train an enhanced language generation model by incorporating the entity relationships to improve the language representation.

# CHAPTER 4

# ENHANCE TEXT GENERATION WITH INTERNAL LINGUISTIC FEATURES

This chapter was previously published as "Enriching Non-Autoregressive Transformer with Syntactic and Semantic Structures for Neural Machine Translation" in EACL'21 (Liu et al., 2021). https://aclanthology.org/2021.eacl-main.105

## 4.1 Introduction

Recently, non-autoregressive models (Gu et al., 2018), which aim to allow the parallel generation of output tokens without losing the translation quality, have attracted much attention. Although the non-autoregressive models have considerably sped up the inference process for real-time neural machine translation (NMT) (Gu et al., 2018), their performance is considerably worse than that of autoregressive counterparts. Most previous works attribute the poor performance to the inevitable conditional independence issue when predicting target tokens, and many variants have been proposed to solve it. For example, several techniques in non-autoregressive models are investigated to mitigate the trade-off between speedup and performance, including iterative refinement (Lee et al., 2018), insertion-based models (Chan et al., 2019; Stern et al., 2019), latent variable based models (Kaiser et al., 2018; Shu et al., 2020), CTC models (Libovický and Helcl, 2018; Saharia et al., 2020), alternative loss function based models (Wei et al., 2019; Wang et al., 2019; Shao et al., 2020), and masked language mod-

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| EN: | A | republican | strategy | to | counter | the | rel-election | of | Obama | . |
| EN POS: | DET | ADJ | NOUN | PART | VERB | DET | NOUN | ADP | PROPN | PUNCT |
| EN NER: | O | B_NORP | O | O | O | O | O | O | B_PERSON | O |
| DE: | Eine | republikanische | strategie | gegen | die | wiederwahl | Obama | . | | |
| DE POS: | DET | ADJ | NOUN | ADP | DET | NOUN | PROPN | PUNCT | | |
| EN NER: | O | B_NORP | O | O | O | O | B_PERSON | O | | |

POS: DET : determiner, ADJ : adjective, NOUN : noun, PART : particle, VERB : verb, ADP : ad-position, PROPN : proper noun, PUNCT : punctuation.
NER: O : object, B_NORP : begin of nationalities, B_PERSON : begin of people

Figure 11. A motivating example on WMT14 En→De dataset. English with POS|NER and its corresponding German translation with POS|NER. The Blue labels show the same tags, while the Red labels show the different tags in two languages.

els (Ghazvininejad et al., 2019; Ghazvininejad et al., 2020). Although these works have tried to narrow the performance gap between autoregressive and non-autoregressive models, and have achieved some improvements on machine translation, the non-autoregressive models still suffer from syntactic and semantic limitations. That is, the translations of non-autoregressive models tend to contain incoherent phrases (e.g., repetitive words), and some informative tokens on the source side are absent. It is because in non-autoregressive models, each token in the target sentence is generated independently. Consequently, it will cause the multimodality issue, i.e., the non-autoregressive models cannot model the multimodal distribution of target sequences properly (Gu et al., 2018).

One key observation to mitigate the syntactic and semantic error is that source and target translated sentences follow the same structure, which can be reflected from Part-Of-Speech (POS) tags and Named Entity Recognition (NER) labels. Briefly, POS, which aims to as-

sign labels to words to indicate their categories by considering the long-distance structure of sentences, can help the model learn the syntactic structure to avoid generating the repetitive words. Likewise, NER, which discovers the proper nouns and verbs of sentences, naturally helps the model recognize some meaningful semantic tokens that may improve translation quality. This observation motivates us to leverage the syntactic as well as semantic structures of natural language to improve the performance of non-autoregressive NMT. We present a motivating example in Figure 11 to better illustrate our idea. From this table, we can find that although the words are altered dramatically from the English sentence to its German translation, the corresponding POS and NER tags still remain similar. For example, most POS tags are identical and follow the same pattern, except that PART, VERB, and ADP in the English do not match the German ADP, while the NER tags are exactly the same in both sentences.

In this paper, we propose an end-to-end Syntactic and semantic structure-aware Non-Autoregressive Transformer model (**SNAT**) for NMT. We take the structure labels and words as inputs of the model. With the guidance of extra sentence structural information, the model greatly mitigates the multimodality issue's negative impact. The core contributions of this paper can be summarized as that we propose 1) a syntax and semantic structure-aware Transformer which takes sequential texts and the structural labels as input and generates words conditioned on the predicted structural labels, and 2) an intermediate alignment regularization which aligns the intermediate decoder layer with the target to capture coarse target information. We conduct experiments on four benchmark tasks over two datasets, including WMT14 En→De and WMT16 En→Ro. In comparison with existing state-of-the-art autoregressive and

non-autoregressive machine translation models, experimental results indicate that our proposed method achieves competitive results as well as significantly reduces the decoding time.

## 4.2 Background

In term of the generation convenience and effectiveness, there are two major problems in the the autoregressive decoding methods. The first problem is that they cannot generate multiple tokens in the same time, which causes the inefficient use of parallel hardware for example GPUs. The second problem is that beam search in the autoregressive method has been found to output low-quality translation with large beam size and even deteriorates when applied to larger search spaces (larger vocabulary). However, non-autoregressive transformer (NAT) could potentially solve these issues. In Particular, the motivation of NAT is through removing the sequential dependencies in the decoding process, it can generate multiple target tokens in one time leading to speed up decoding process, which is demonstrated by the following equation:

$$P_{\mathbf{NAR}}(\mathbf{y}|\mathbf{x};\phi) = \prod_{t=1}^{m} p\left(y_t|\hat{\mathbf{x}},\mathbf{x};\phi\right), \tag{4.1}$$

where $\hat{\mathbf{x}} = \{\hat{x}_1,\ldots,\hat{x}_m\}$ is the copied source sentence. Because the conditional dependencies in the generated target sentence are removed ($y_t$ doesn't depends on $y_{<t}$), the decoder cannot utilize the inherent target semantic information for token prediction. Therefore, during training stage, the decoder is supposed to find out such target-side semantic information by itself just laying on the source-side semantic information. This is a much more difficult task compared to the autoregressive ones. From our investigation, we find the NAT models fail to handle

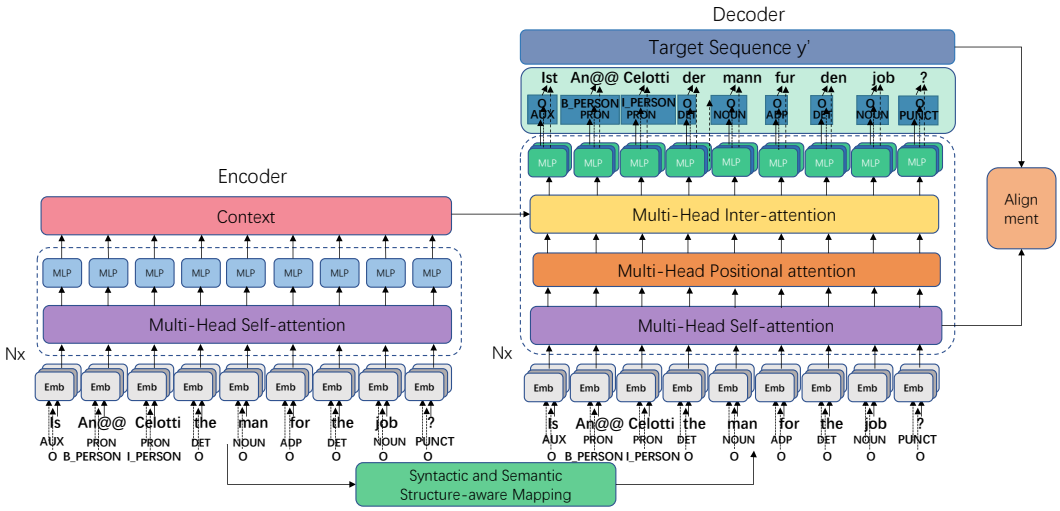



Figure 12. An overview of the proposed **SNAT** for neural machine translation.

the target sentence generation well. It usually generates repetitive and semantically incoherent sentences with missing words. Therefore, strong conditional signals should be considered as the decoder input to help the decoder better learn internal dependencies within the generated sentence.

## 4.3 Methodology

In this section, we present our model **SNAT** to incorporate the syntactic and semantic structure information into a NAT model as well as an intermediate latent space alignment within the target. Figure 12 gives an overview of the network structure of our proposed **SNAT**. In **SNAT**, the input sequence is segmented into sub-words by byte-pair tokenizer (Sennrich et al., 2016). In parallel, words in the input sequence are passed to POS and NER annotators to extract explicit syntactic and semantic structures, and the corresponding embeddings are

aggregated by a linear layer to form the final syntax and semantic structure-aware embedding. The **SNAT** model copies the structured encoder input as the decoder input and generates the translated sentences and labels.

One of the most important properties of **SNAT** is that it naturally introduces syntactic and semantic information when taking the structure-aware information as inputs and generating both words and labels. More precisely, given a source sentence $\mathbf{x}$, as well as its label sequence $\mathbf{L_x}$, the conditional probability of a target translation $\mathbf{y}$ and its label sequence $\mathbf{L_y}$ is:

$$\begin{aligned} & P_{\mathbf{SNAT}}(\mathbf{y}, \mathbf{L_y} | \mathbf{x}, \mathbf{L_x}; \varphi) \\ = & \prod_{t=1}^{m} p\left(y_t, L_{y_t} | \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L_x}; \varphi\right), \end{aligned} \tag{4.2}$$

where $\mathbf{x}$ and $\mathbf{L_x}$ are first fed into the encoder of **SNAT** model. $\hat{\mathbf{x}}$ and $\hat{\mathbf{L}}_{\mathbf{x}}$ with length $m$ are syntactic and semantic structure-aware copying of word and label from encoder inputs, respectively. We show the details in the following sections.

### 4.3.1 Syntactic and Semantic Labeling

We use POS and NER to introduce the syntactic and semantic information existing in natural language, respectively. During the data pre-processing, each sentence is annotated into a semantic sequence using an open-source pre-trained semantic annotator. In particular, we take the Treebank style (Marcus et al., 1999) for POS and PropBank style (Palmer et al., 2005) for NER to annotate every input sequence token with the semantic labels. Given a specific sentence, there would be its corresponding predicate-argument structures. Since the input sequence is segmented into subword units using byte-pair encoding (Sennrich et al., 2016), we

assign the same label to all subwords tokenized from the same word. As shown in Figure 12, the word "*Ancelotti*" is tokenized as "*An@@*" and "*Celotti*". The corresponding POS tags are PRON and PRON while the corresponding NER tags are B_PERSON and I_PERSON. For the text "*Is An@@ Celotti the man for the job ?*", the corresponding POS tag set is {AUX, PRON, PRON, DET, NOUN, ADP, DET, NOUN, PUNCT} and the NER tag set is {O, B_PERSON, I_PERSON, O, O, O, O, O, O}. The data flow of the proposed model is also shown in Figure 12.

### 4.3.2 Encoder

As same as Transformer (Vaswani et al., 2017), we design the encoder as a stack of six identical multi-head attention blocks. In addition to the word embedding and position embedding in the traditional Transformer, we add structure-aware label embedding. The input to the encoder block is the addition of the normalized word, labels (NER and POS) and position embedding, which is represented as $\mathbf{H}^0 = [\mathbf{h}_1^0, \ldots, \mathbf{h}_n^0]$.

The input representation in the first layer $\mathbf{H}^0 = [\mathbf{h}_1^0, \ldots, \mathbf{h}_n^0]$ is encoded as contextual representations through the stacked multi-head attention blocks. In each layer, the contextual layer representation $\mathbf{H}^l = [\mathbf{h}_1^l, \ldots, \mathbf{h}_n^l]$ is computed by the $l$-th layer Transformer encoder block $\mathbf{H}^l = \text{Transformer}_l(\mathbf{H}^{l-1})$, $l \in \{1, 2, \ldots, 6\}$. In each Transformer encoder block, the previous layer output vectors are updated by the multiple self-attention heads by considering the attention of the contextual information. This self-attention mechanism can be represented as the weighted sum of the similarity between value vector $\mathbf{V}$ and the query vector $\mathbf{Q}$ times the key vector $\mathbf{K}$:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{\text{model}}}}\right) \cdot \mathbf{V}, \tag{4.3}$$

where $d_{\text{model}}$ indicates hidden representation dimension. In the self-attention, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are mappings of previous hidden representation by different linear functions, i.e., $\mathbf{Q} = \mathbf{H}^{l-1}\mathbf{W}_Q^l$, $\mathbf{K} = \mathbf{H}^{l-1}\mathbf{W}_K^l$, and $\mathbf{V} = \mathbf{H}^{l-1}\mathbf{W}_V^l$, respectively. At last, the encoder produces a final contextual representation $\mathbf{H}^6 = [\mathbf{h}_1^6, \ldots, \mathbf{h}_n^6]$, which is obtained from the last Transformer block.

### 4.3.3 Decoder

The decoder consists of six identical Transformer decoder blocks, but with several main differences from the encoder. More specifically, we denote the contextual representations in the $i$-th decoder layer is $\mathbf{Z}^i (1 \leq i \leq 6)$. The input to the decoder block as $\mathbf{Z}^0 = [\mathbf{z}_1^0, \ldots, \mathbf{z}_m^0]$, which is produced by the addition of the word, labels (NER and POS) copying from encoder input and positional embedding.

For the target side input $[\hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}]$, most previous works simply copied partial source sentence with the length ratio $\frac{n}{m}$ where $n$ refers to the source length and $m$ is the target length as the decoder input. More detail, the decoder input $y_i$ at the $i$-th position is a copy of the $\lfloor \frac{n}{m} \times i \rfloor$th contextual representation, i.e., $x_{\lfloor \frac{n}{m} \times i \rfloor}$ from the encoder. From our investigation, in most cases, the gap between source length and target length is relatively small (e.g. 2). Therefore, it deletes or duplicates the copy of the last a few tokens of the source. If the last token is meaningful, the deletion will neglect important information. Otherwise, if the last token is trivial, multiple copies will add noise to the model.

Instead, we propose a syntactic and semantic structure-aware mapping method considering the POS and NER labels to construct the decoder inputs. Our model first picks out the informative words with NOUN and VERB POS tags, and the ones recognized as entities by the

NER module. If the source length is longer than the target length, we retain all informative words, and randomly delete the rest of the words. On the other hand, if the source length is shorter than the target, we retain all words and randomly duplicate the informative words. The corresponding label of a word is also deleted or preserved. Moreover, by copying the similar structural words from the source, it can provide more information to the target input than just copying the source token, which is greatly different from the target token. The POS and NER labels of those structure-aware copied words from the source sentence are also copied as the decoder input. So by using the structure-aware mapping, we can assign $[\hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}]$ as decoder input. For positional attention which aims to learn the local word orders within the sentence (Gu et al., 2018), we set positional embedding (Vaswani et al., 2017) as both $\mathbf{Q}$ and $\mathbf{K}$, and the hidden representations of the output from the previous layer as $\mathbf{V}$. For inter-attention, $\mathbf{Q}$ denotes hidden representations of the previous layer. $\mathbf{K}$ and $\mathbf{V}$ are contextual last layer vectors $\mathbf{H}^6$ from the encoder. We modify the attention mask in original Transformer so that it does not mask out the future tokens, and every token is dependent on both its preceding and succeeding tokens in every layer. Therefore, the generation of each token can use bi-directional attention. The position-wise Feed-Forward Network (FFN) is implemented after multi-head attention in both encoder and decoder. It is consisted by two fully-connected layers and a layer normalization (Ba et al., 2016). The FFN takes $\mathbf{Z}^6$ as input and calculates the final representation $\mathbf{Z}^f$, which is used to predict the whole target sentence and label:

$$p\left(\mathbf{y} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L}_{\mathbf{x}}\right) = f\left(\mathbf{Z}^f \mathbf{W}_w^\top + \mathbf{b}_w\right), \tag{4.4}$$

$$q\left(\mathbf{L_y} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}\right) = f\left(\mathbf{Z}^f \mathbf{W}_l^\top + \mathbf{b}_l\right), \tag{4.5}$$

where $f$ is a GeLU activation function (Hendrycks and Gimpel, 2016b). $\mathbf{W}_w$ and $\mathbf{W}_l$ are the token embedding and structural label embedding in the input representation, respectively. We use different FFNs for POS and NER labels. To avoid redundancy, we just use $q\left(\mathbf{L_y} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}\right)$ to represent the two predicted label likelihood in general.

### 4.3.4 Training

We use ($\mathbf{x}$, $\mathbf{L_x}$, $\mathbf{y}^*$, $\mathbf{L}_\mathbf{y}^*$) to denote a training instance. To introduce the label information, our proposed **SNAT** contains a discrete sequential latent variable $L_{y_{1:m}}$ with conditional posterior distribution $p(L_{y_{1:m}}|\hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}; \varphi)$. It can be approximated using a proposal distribution $q(\mathbf{L_y} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x})$. The approximation also provides a variational bound for the maximum log-likelihood:

$$\begin{aligned}
&\log P_{\mathbf{SNAT}} = \log \sum_{t=1}^{m} q\left(L_{y_t}|\hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}; \varphi\right) \\
&\times p\left(y_t|L_{y_t}, \hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}; \varphi\right) \\
&\geq \mathop{\mathbb{E}}_{L_{y_{1:m}} \sim q} \Big\{ \underbrace{\sum_{t=1}^{m} \log q\left(L_{y_t} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}; \varphi\right)}_{\text{Label likelihood}} \\
&+ \underbrace{\sum_{t=1}^{m} \log p\left(y_t \mid L_{y_t}, \hat{\mathbf{x}}, \hat{\mathbf{L}}_\mathbf{x}, \mathbf{x}, \mathbf{L_x}; \varphi\right)}_{\text{Structure-aware word likelihood}} \Big\} + \mathcal{H}(q).
\end{aligned} \tag{4.6}$$

Note that, the resulting likelihood function, consisting of the two bracketed terms in Equation 4.6, allows us to train the entire model in a supervised fashion. The inferred label can be

utilized to train the label predicting model $q$ and simultaneously supervise the structure-aware word model $p$. The label loss can be calculated by the cross-entropy $\mathcal{H}$ of $L^*_{y_t}$ and Equation 4.5):

$$\mathcal{L}_{\text{label}} = \sum_{t=1}^{m} \mathcal{H}\left(L^*_{y_t}, \quad q(L_{y_t} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L}_{\mathbf{x}})\right), \tag{4.7}$$

The structure-aware word likelihood is conditioned on the generation result of the label. Since the Equation 4.4 does not depend on the predicted label, we propose to bring the structure-aware word mask $\mathbf{M}_{wl} \in \mathbb{R}^{|V_{word}| \times |V_{label}|}$, where $|V_{word}|$ and $|V_{label}|$ are vocabulary sizes of word and label, respectively. The mask $\mathbf{M}_{w_l}$ is defined as follows:

$$\mathbf{M}_{w_l}(i,j) = \begin{cases} 1, & \mathcal{A}(y_i) = label_j, \\ \epsilon, & \mathcal{A}(y_i) \neq label_j, \end{cases} \tag{4.8}$$

which can be obtained at the preprocessing stage, and $\mathcal{A}$ denotes the open-source pre-trained POS or NER annotator mentioned above. It aims to penalize the case when the word $y_i$ does not belong to the label $label_j$ with $\epsilon$, which is a small number defined within the range of $(0, 1)$ and will be tuned in our experiments. For example, the word "great" does not belong to VERB. The structure-aware word likelihood can be reformulated as:

$$\begin{aligned} p(y_t \mid L_{y_t}, \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L}_{\mathbf{x}}; \varphi) &= p(y_t \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L}_{\mathbf{x}}) \\ \times \mathbf{M}_{w_l} &\times q(L_{y_t} \mid \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L}_{\mathbf{x}}). \end{aligned} \tag{4.9}$$

Consequently, the structure-aware word loss $\mathcal{L}_{\text{word}}$ is defined as the cross-entropy between true $p^{'}(y_t^*|L_{y_t}^*)$ and predicted $p(y_t \mid L_{y_t}, \hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}, \mathbf{x}, \mathbf{L}_{\mathbf{x}}; \varphi)$, where $p^{'}(y_t^*|L_{y_t}^*) \in \mathbb{R}^{|V_{word}|\times|V_{label}|}$ is a matrix where only item at the index of $(y_t^*, L_{y_t}^*)$ equals to 1, otherwise equals to 0. We reshape $p^{'}(y_t^*|L_{y_t}^*)$ and $p(y_t|L_{y_t})$ to vectors when calculating the loss. **Intermediate Alignment Regularization** One main problem of NAT is that the decoder generation process does not depend on the previously generated tokens. Based on the bidirectional nature of **SNAT** decoder, the token can depend on every token of the decoder input. However, since the input of decoder $[\hat{\mathbf{x}}, \hat{\mathbf{L}}_{\mathbf{x}}]$ is the duplicate of encoder input $[\mathbf{x}, \mathbf{L}_{\mathbf{x}}]$, the generation depends on the encoder tokens rather than the target $\mathbf{y}^*$.

To solve this problem, we align the output of the intermediate layer of the decoder with the target. The alignment makes the generation of following layers dependent on the coarse target-side information instead of the mere encoder input. This alignment idea is inspired by (Guo et al., 2019), which directly feeds target-side tokens as inputs of the decoder by linearly mapping the source token embeddings to target embeddings. However, using one FFN layer to map different languages to the same space can hardly provide promising results. Thus, instead of aligning the input of decoder with the target, we use the intermediate layer of decoder to align with the target. In this case, our model avoids adding additional training parameters and manages to train the alignment together with **SNAT** model in an end-to-end fashion. Formally, we define the intermediate alignment regularization as cross-entropy loss between the predicted word and the true word:

$$\mathcal{L}_{\text{reg}} = \sum_{t=1}^{m} \mathcal{H}\left(y_t^*, \quad \text{FFN}(\mathbf{Z}_t^{md})\right), \tag{4.10}$$

where $\mathbf{Z}^{md}$ $(1 < md < 6)$ represents the output of each intermediate layer. Consequently, the final loss of **SNAT** can be represented with the coefficient $\lambda$ as:

$$\mathcal{L}_{\mathbf{SNAT}} = \mathcal{L}_{\text{word}} + \mathcal{L}_{\text{label}} + \lambda \mathcal{L}_{\text{reg}}. \tag{4.11}$$

## 4.4 Experiment

In this section, we conduct experiments to evaluate the effectiveness and efficiency of our proposed model, with comprehensive analysis.

### 4.4.1 Experimental Setup

**Data**

We evaluate **SNAT** performance on both the WMT14 En-De, which contains of around 4.5M sentence pairs and the WMT16 En-Ro, which contains of around 610k sentence pairs in parallel corpora. In order to be consistent with the previous publications, we processed the data following (Ghazvininejad et al., 2019) for the parallel data. The dataset is processed with Moses script (Hoang and Koehn, 2008), and the words are segmented into subword units using byte-pair encoding (Sennrich et al., 2016). The WMT14 En-De task uses newstest-2013 and newstest-2014 as dev and test sets, and WMT16 En-Ro task uses newsdev-2016 and newstest-2016 as dev and test sets. We report all results on test sets. The vocabulary is shared between source and target languages and has ∼36k units and ∼34k units in WMT14 En-De and WMT16 En-Ro, respectively.

**Model Configuration**

| | En→De | De→En | En→Ro | Ro→En | | |
|---|---|---|---|---|---|---|
| ***Autoregressive Models*** | | | | | **Latency** | **Speedup** |
| **LSTM Seq2Seq** (Bahdanau et al., 2017) | 24.60 | - | - | - | - | - |
| **Conv S2S** (Gehring et al., 2017) | 26.43 | - | 30.02 | - | - | - |
| **Transformer**† (Vaswani et al., 2017) | 27.48 | 31.29 | 34.36 | 33.82 | 642ms | 1.00X |
| ***Non-autoregressive Models*** | | | | | **Latency** | **Speedup** |
| **NAT** (Gu et al., 2018) | 17.69 | 20.62 | 29.79 | - | 39ms | 15.6X |
| **NAT, rescoring 10** (Gu et al., 2018) | 18.66 | 22.41 | - | - | 79ms | 7.68X |
| **NAT, rescoring 100** (Gu et al., 2018) | 19.17 | 23.20 | - | - | 257ms | 2.36X |
| **iNAT** (Lee et al., 2018) | 21.54 | 25.43 | 29.32 | - | - | 5.78X |
| **Hint-NAT** (Li et al., 2020) | 21.11 | 25.24 | - | - | 26ms | 23.36X |
| **FlowSeq-base** (Ma et al., 2019) | 21.45 | 26.16 | - | 29.34 | - | - |
| **ENAT-P** (Guo et al., 2019) | 20.26 | 23.23 | 29.85 | - | 25ms | 24.3X |
| **ENAT-P, rescoring 9** | 23.22 | 26.67 | 34.04 | - | 50ms | 12.1X |
| **ENAT-E** | 20.65 | 23.02 | 30.08 | - | 24ms | 25.3X |
| **ENAT-E, rescoring 19** | 24.28 | 26.10 | 34.51 | - | 49ms | 12.4X |
| **DCRF-NAT** (Sun et al., 2019b) | 23.44 | 27.22 | - | - | 37ms | 16.4X |
| **DCRF-NAT, rescoring 9** | 26.07 | 29.68 | - | - | 63ms | 6.1X |
| **DCRF-NAT, rescoring 19** | 26.80 | 30.04 | - | - | 88ms | 4.4X |
| **NAR-MT(rescoring 11)** (Zhou and Keung, 2020) | 23.57 | 29.01 | 31.21 | 32.06 | - | - |
| **NAR-MT(rescoring 11) + monolingual** | 25.53 | 29.96 | 31.91 | 33.46 | - | - |
| **AXE CMLM** (Ghazvininejad et al., 2020) | 23.53 | 27.90 | 30.75 | 31.54 | - | - |
| **SNAT** | 24.64 | 28.42 | 32.87 | 32.21 | 26.88ms | 22.6X |
| **SNAT, rescoring 9** | 26.87 | 30.12 | 34.93 | 33.11 | 54.63ms | 11.1X |
| **SNAT, rescoring 19** | **27.50** | **30.82** | **35.19** | **33.98** | 65.62ms | 9.3X |

TABLE XI. Performance of BLEU score on WMT14 En↔De and WMT16 En↔Ro tasks.

Our implementation is based on the PyTorch sequence modeling toolkit Fairseq.[1] We follow the weights initialization scheme from BERT and follow the settings of the Transformer base version configuration in (Vaswani et al., 2017) for all the models. In detail, we use six layers per stack, eight attention heads per layer, 512 model dimensions and 2,048 hidden dimensions. The dimension of POS and NER embedding is set to 512 which is the same as the word embedding dimension. The autoregressive and non-autoregressive model have the similar structure, except for the encoder input, decoder attention mask and the decoding input for the non-autoregressive

[1] https://github.com/pytorch/fairseq

model as we described in Sec. 4.3. We try different values for the label mismatch penalty $\epsilon$ from $\{0.01, 0.1, 0.5\}$ and find that 0.1 gives the best performance. The coefficient $\lambda$ is tested with different values from $\{0.25, 0.5, 0.75, 1\}$, and $\lambda = 0.75$ outperforms other settings. We set the initial learning rate as values from {1e-6, 1e-5, 2e-5, 3e-5}. The warm-up rate is set as 0.1 and L2 weight decay is set as 0.01. Sentences are tokenized and the maximum number of tokens in each step is set to 8,000. The maximum iteration step is set to 30,000, and we train the model with early stopping.

**Baselines** We choose the following models as baselines: **NAT** is a vanilla non-autoregressive Transformer model for NMT which is first introduced in (Gu et al., 2018). **iNAT** (Lee et al., 2018) improves the vanilla NAT model through reading and refining iteratively in the translation process. The number of iterations is usually set to 10 for decoding. **Hint-NAT** (Li et al., 2020) utilizes the intermediate hidden states from an autoregressive teacher to improve the NAT model. **FlowSeq** (Ma et al., 2019) adopts normalizing flows (Kingma and Dhariwal, 2018), which used as latent variables for generation. **ENAT** (Guo et al., 2019) proposes two ways to enhance the decoder inputs to improve NAT models. The first one leverages a phrase table to translate source tokens to target tokens **ENAT-P**. The second one transforms source-side word embedding into target-side word embeddings **ENAT-E**. **DCRF-NAT** (Sun et al., 2019b) designs an approximation of conditional random field on the top of NAT models and further uses a dynamic transition methodology to represent the positional context in the conditional random field. **NAR-MT** (Zhou and Keung, 2020) uses a large number of monolingual corpora source texts to generate additional teacher outputs. **AXE CMLM** (Ghazvininejad et al., 2020)

trains the conditional masked language models using a differentiable dynamic program to assign the training loss. It is gives more reward to the best possible monotonic alignment between ground-truth tokens and model predictions.

### 4.4.2 Training and Inference Details

To obtain the part-of-speech and named entity labels, we use industrial-strength spaCy[1] to acquire the label for English, German, and Romanian input. In our implementation, there are 17 labels for POS in total, i.e., ADJ (adjective), ADV (adverb), ADP (ad-position), AUX (auxiliary), CCONJ (coordinating conjunction), DET (determiner), INTJ (interjection), NOUN (noun), NUM (numeral), PART (particle), PRON (pronoun), PROPN (proper noun), PUNCT (punctuation), SCONJ (subordinating conjunction), SYM (symbol), VERB (verb), and X (other). The NER task is trained on OntoNotes v5.0 benchmark dataset (Pradhan et al., 2013) using formatted BIO labels and defines 18 entity types: CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, ORDINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME, and WORK_OF_ART.

**Knowledge Distillation** Similar to previous works on non-autoregressive translation (Gu et al., 2018; Ghazvininejad et al., 2019; Shu et al., 2020), we adopt the knowledge distillation of the candidate generated sentences of the different length using a standard left-to-right Transformer model (i.e., Transformer-large for WMT14 EN→DE, and Transformer-base for WMT16 EN→RO) to get score for each candidates. Specifically, we use scaling NMT (Ott et al., 2018) as

---

[1] `https://spacy.io/usage/models`

the teacher model. We report the performance of standard autoregressive Transformer trained on distilled data for WMT14 EN→DE and WMT16 EN→RO. We average the last five training checkpoints to obtain the final model. We train the model with cross-entropy loss and label smoothing ($\epsilon = 0.1$).

**Inference** During training, we do not need to predict the target length $m$ since the target sentence is given. During inference, we use a simple method to select the target length for **SNAT** (Wang et al., 2019; Li et al., 2020). First, we put the target length to $m' = n + C$, where $n$ is the length of the source sentence and $C$ is a constant bias term estimated from the overall length statistics of the training data. Then, we create a list of candidate target lengths with a range of $[m' - B, m' + B]$ where $B$ is the half-width of the interval. Finally, the model picks the best one from the generated $2B + 1$ candidate sentences. In our experiments, we set the constant bias term $C$ to 2 for WMT 14 EN→DE, -2 for WMT 14 DE→EN, 3 for WMT 16 EN→RO, and -3 for WMT 14 RO→EN based on the average lengths of different languages in the training sets. We set $B$ to 4 or 9, and obtain corresponding 9 or 19 candidates for each translated sentence. Then we employ an autoregressive teacher model to rescore these candidates.

### 4.4.3 Results and Analysis

The experiment results are shown in Table XI. In terms of translation quality, we compare the structure-aware non-autoregressive method with autoregressive models under the metrics of BLEU score (Papineni et al., 2002). For all datasets, we obtain comparable results with the Transformer, the state-of-the-art autoregressive model. Our best model achieves 27.50

(+0.02 gain over Transformer), 30.82 (-0.46 gap with Transformer), 35.19 (+0.82 gain), and 33.98 (+0.16 gain) BLEU score on WMT14 En↔De and WMT16 EN↔Ro, respectively. More importantly, our **SNAT** can decode much faster than the autoregressive Transformer, which is a big improvement regarding the speed-accuracy trade-off.

In the comparison of our models with other NAT models, we observe that using the best configs of **SNAT** model can get a significant performance improvement over the models NAT, iNAT, Hint-NAT, ENAT, FlowSeq, NAR-MT and AXE CMLM by +8.33, +5.96, +6.39, +6.05, +3.22, 3.93 and +3.97 in BLEU on WMT14 En→De, respectively. This indicates that the incorporation of the syntactic and semantic structure largely reduces the impact of the multimodality problem. Thus it narrows the performance gap between Autoregressive Transformer (AT) and Non-Autoregressive Transformer (NAT) models. In addition, we see a +0.69, +0.78, +0.68, and 0.52 gain of BLEU score over the best baselines on WMT14 En→De, WMT14 De→En, WMT16 En→Ro and WMT16 Ro→En, respectively.

Based on the result of our methods at the last group in Table XI, we find that the rescoring technique substantially assists in boosting the performance. Specifically, on En→De, rescoring 9 candidates results in a gain of +2.23 BLEU, and rescoring 19 candidates gets a +2.86 BLEU score increment.

**Decoding Speed** Following previous works (Gu et al., 2018; Lee et al., 2018; Guo et al., 2019), we evaluate the average latency of each sentence decoding on WMT14 En→De test sets with the batch size being 1, under an environment of NVIDIA Titan RTX GPU for the Transformer model and the NAT models to measure the latency cost. The reported latencies are

attained by taking an average of five runs. More clearly, We reproduce the Transformer on our machine. We copy the runtime of other models but the speedup ratio is between the runtime of their implemented Transformer and their proposed model. We think it's reasonable to compare the speedup ratio because it is independent of the influence caused by different implementation software or machines. And to clarify, the latency does not include preprocessing of tagging, because it's a very fast process as executing around 7000 sentences in one second.

We can see from Table XI that the best **SNAT** gets a 9.3 times decoding speedup than the Transformer, while achieving comparable or even better performance. Compared to other NAT models, we observe that the **SNAT** model is nearly the fastest which is just slightly behind of ENAT and Hint-NAT with regard to latency, and is significantly faster than DCRF-NAT with the better performance.

| **Model** | **POS tag** | **NER tag** | **BLEU** |
|---|---|---|---|
| SNAT-V1 | ✔ | | 24.21 |
| SNAT-V2 | | ✔ | 24.09 |
| SNAT-V3 | | | 22.84 |

TABLE XII. The performance of different vision of SNAT models on WMT14 En→De development set. ✔ means selecting the label tag.

| Method | WMT14 En→ De | WMT14 De→ En |
|---|---|---|
| w/o | 23.11 | 27.03 |
| w/ $\mathbf{Z}^2$ | 24.32 | 28.21 |
| w/ $\mathbf{Z}^3$ | 24.57 | 28.42 |

TABLE XIII. The performance with respect to using different layer of intermediate interaction. Evaluated by the BLEU score on WMT14 En→De|WMT14 De→En.

| Model | 10 | 20 | 30 | 50 | 100 |
|---|---|---|---|---|---|
| AT | 28.35 | 28.32 | 28.30 | 24.26 | 20.73 |
| NAT | 21.31 | 19.55 | 17.19 | 16.31 | 11.35 |
| SNAT | 28.67 | 28.50 | 27.33 | 25.41 | 17.69 |

TABLE XIV. The performance with respect to different sentence lengths. Evaluated by the BLEU score on WMT14 En→De.

#### 4.4.4 Ablation Analysis

**Effect of Syntactic and Semantic Structure Information** We investigate the effect of using the syntactic and semantic tag on the model performance. Experimental results are shown in Table XII. It demonstrates that incorporating POS information boosts the translating performance (+1.37 on WMT14 En→De) and NER information can also enhance the translating performance (+1.25 on WMT14 En→De). The POS label enriches the model with the syntactic structure, while the NER label supplements the semantic information to the model which are critical elements for **SNAT** model to exhibit better translation performance.

**Effect of Intermediate Representation Alignment** We conduct experiments for our **SNAT** model on WMT14 En→De with various alignments between decoder layers and target.

As shown in Table XIII, using the second layer $\mathbf{Z}^2$ in the decoder as intermediate alignment can gain +1.21 improvement, while using the third layer $\mathbf{Z}^3$ in the decoder as intermediate alignment can gain +1.46 improvement. This is as same as our expectation that by aggregating layer-wise token information in the intermediate layers can assist to capture the generated sentence dependencies so that enhance the decoder's ability.

**Effect of Sentence Length** To evaluate the performance of different models on different sentence lengths, we conduct experiments on the WMT14 En→De development set. Based on the length of the reference sentences, We divide the sentence pairs into different length buckets. As shown in Table XIV, the column of 100 calculates the BLEU score of sentences that the length of the reference sentence is larger than 50 but smaller or equal to 100. We can see that the performance of vanilla NAT drops quickly as the sentence length increases from 10 to 50, while AT model and the proposed **SNAT** model have relatively stable performance over different sentence lengths. This result verifies the ability of the proposed model in learning the long-term token dependencies.

# CHAPTER 5

# ENHANCE TEXT GENERATION WITH INTERNAL GRAPH STRUCTURE

This chapter was previously published as "HETFORMER: Heterogeneous Transformer with Sparse Attention for Long-Text Summarization" in EMNLP'21 (Liu et al., 2021). https://arxiv.org/abs/2110.06388

## 5.1 Introduction

Recent years have seen a resounding success in the use of graph neural networks (GNNs) on document summarization tasks (Wang et al., 2020a; Hanqi Jin, 2020), due to their ability to capture inter-sentence relationships in complex document.

Since GNN requires node features and graph structure as input, various methods, including extraction and abstraction (Li et al., 2020a; Huang et al., 2020; Jia et al., 2020), have been proposed for learning desirable node representations from raw text. Particularly, they have shown that Transformer-based pre-trained language models such as BERT and RoBERTa (Devlin et al., 2018; Liu et al., 2019b) offer an effective way to initialize and fine tune the node representations as the input of GNN.

Despite great success in combining Transformer-based pre-trained models with GNNs, all existing approaches have their limitations. The first limitation lies in the adaptation capability to long-text input. Most pre-trained methods truncate longer documents into a small fixed-

length sequence (e.g., $n = 512$ tokens), as its attention mechanism requires a quadratic cost w.r.t. sequence length. This would lead to serious information loss (Li et al., 2020a; Huang et al., 2020). The second limitation is that they use pre-trained models as a multi-layer feature extractor to learn better node features and build multi-layer GNNs on top of extracted features, which have cumbersome networks and tremendous parameters (Jia et al., 2020).

Recently there have been several works focusing on reducing the computational overhead of fully-connected attention in Transformers. Especially, ETC (Ravula et al., 2020) and Longformer (Beltagy et al., 2020) proposed to use local-global sparse attention in pre-trained models to limit each token to attend to a subset of the other tokens (Child et al., 2019), which achieves a linear computational cost of the sequence length. Although these methods have considered using local and global attentions to preserve hierarchical structure information contained in raw text data, their abilities are still not enough to capture multi-level granularities of semantics in complex text summarization scenarios.

In this work, we propose HetFormer, a HEterogeneous transFORMER-based pre-trained model for long-text summarization using multi-granularity sparse attentions. Specifically, we treat tokens, entities, sentences as different types of nodes and the multiple sparse masks as different types of edges to represent the relations (e.g., token-to-token, token-to-sentence), which can preserve the graph structure of the document even with the raw textual input. Moreover, our approach will eschew GNN and instead rely entirely on a sparse attention mechanism to draw heterogeneous graph structural dependencies between input tokens.

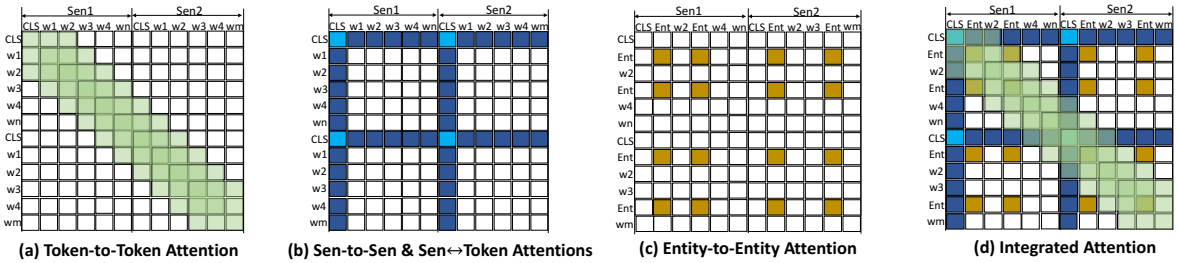



Figure 13. An illustration of sparse attention patterns ((a), (b), (c)) and their combination (d) in HetFormer.

The main contributions of the paper are summarized as follows: 1) we propose a new structured pre-trained method to capture the heterogeneous structure of documents using sparse attention; 2) we extend the pre-trained method to longer text summarization instead of truncating the document to small inputs; 3) we empirically demonstrate that our approach achieves state-of-the-art performance on both single- and multi-document summarization tasks.

## 5.2 HetFormer on Summarization

HetFormer aims to learn a heterogeneous Transformer in pre-trained model for text summarization. To be specific, we model different types of semantic nodes in raw text as a potential heterogeneous graph, and explore multi-granularity sparse attention patterns in Transformer to directly capture heterogeneous relationships among nodes. The node representations will be interactively updated in a fine-tuned manner, and finally, the sentence node representations are used to predict the labels for text summarization.

### 5.2.1 Node Construction

In order to accommodate multiple granularities of semantics, we consider three types of nodes: *token*, *sentence* and *entity*.

The *token* node represents the original textual item that is used to store token-level information. Different from HSG (Wang et al., 2020a) which aggregates identical tokens into one node, we keep each token occurrence as a different node to avoid ambiguity and confusion in different contexts. Each *sentence* node corresponds to one sentence and represents the global information of one sentence. Specifically, we insert an external `[CLS]` token at the start of each sentence and use it to encode features of each tokens in the sentence. We also use the interval segment embeddings to distinguish multiple sentences within a document, and the position embeddings to display monotonical increase of the token position in the same sentence. The *entity* node represents the named entity associated with the topic. The same entity may appear in multiple spans in the document. We utilize NeuralCoref[1] to obtain the coreference resolution of each entity, which can be used to determine whether two expressions (or "mentions") refer to the same entity.

### 5.2.2 Sparse Attention Patterns

Our goal is to model different types of relationships (edges) among nodes, so as to achieve a sparse graph-like structure directly. To this end, we leverage multi-granularity sparse attention mechanisms in Transformer, by considering five attention patterns, as shown in Fig-

---

[1] `https://github.com/huggingface/neuralcoref`

ure 13: *token-to-token* (*t2t*), *token-to-sentence* (*t2s*), *sentence-to-token* (*s2t*), *sentence-to-sentence* (*s2s*) and *entity-to-entity* (*e2e*).

Specifically, we use a fixed-size window attention surrounding each token (Fig. 1(a)) to capture the short-term *t2t* dependence of the context. Even if each window captures the short-term dependence, by using multiple stacked layers of such windowed attention, it could result in a large receptive field (Beltagy et al., 2020). Because the top layers have access to all input locations and have the capacity to build representations that incorporate information across the entire input.

The *t2s* represents the attention of all tokens connecting to the sentence nodes, and conversely, *s2t* is the attention of sentence nodes connecting to all tokens across the sentence (the dark blue lines in Fig. 1(b)). The *s2s* is the attention between multiple sentence nodes (the light blue squares in Fig. 1(b)). To compensate for the limitation of *t2t* caused by using fixed-size window, we allow the *sentence* nodes to have unrestricted attentions for all these three types. Thus tokens that are arbitrarily far apart in the long-text input can transfer information to each other through the *sentence* nodes.

Complex topics related to the same entity may appear in multiple sentences, making it difficult for existing sequential language models to fully detain the semantics among entities. To solve this problem, we introduce the *e2e* attention pattern Fig. 1(c). The intuition is that if there are several mentions of a particular entity, all the pairs of the same mentions are connected. In this way, we can facilitate the connections of relevant entities and preserve global context, e.g., entity interactions and topic flows.

**Linear Projections for Sparse Attention.** In order to ensure the sparsity of attention, we create three binary masks for each attention patterns $\mathbf{M}^{t2t}$, $\mathbf{M}^{ts}$ and $\mathbf{M}^{e2e}$, where 0 means disconnection and 1 means connection between pairs of nodes. In particular, $\mathbf{M}^{ts}$ is used jointly for *s2s*, *t2s* and *s2t*. We use different projection parameters for each attention pattern in order to model the heterogeneity of relationships across nodes. To do so, we first calculate each attention with its respective mask and then sum up these three attentions together as the final integrated attention (Fig. 1(d)).

Each sparse attention is calculated as: $\mathbf{A}^m = \text{softmax}\left(\frac{\mathbf{Q}^m \mathbf{K}^{m\top}}{\sqrt{d_k}}\right)\mathbf{V}^m$, $m \in \{t2t, ts, e2e\}$. The query $\mathbf{Q}^m$ is calculated as $(\mathbf{M}^m \odot \mathbf{X})\mathbf{W}_Q^m$, where $\mathbf{X}$ is the input text embedding, $\odot$ represents the element-wise product and $\mathbf{W}_Q^m$ is the projection parameter. The key $\mathbf{K}^m$ and the value $\mathbf{V}^m$ are calculated in a similar way as $\mathbf{Q}^m$, but with respect to different projection parameters, which are helpful to learn better representation for heterogeneous semantics.

The expensive operation of full-connected attention is $\mathbf{QK}^T$ as its computational complexity is related to the sequence length (Kitaev et al., 2020). While in HetFormer, we follow the implementation of Longformer that only calculates and stores attention at the position where the mask value is 1 and this results in a linear increase in memory use compared to quadratic increase for full-connected attention.

### 5.2.3 Sentence Extraction

As summarization is more general and widely used, we build a classifier on each *sentence* node representation $o_s$ to select sentences from the last layer of HetFormer. The binary classifier employs a linear projection layer with the sigmoid activation function to get the

prediction score for each sentence: $\tilde{y}_s = \sigma\left(o_s \mathbf{W}_o + b_o\right)$, where $\sigma$ is the sigmoid function, $\mathbf{W}_o$ and $b_o$ are parameters of projection layer.

In the training stage, these prediction scores are trained learned on the binary cross-entropy loss with the golden labels $y$. In the inference stage, these scores are used to sort the sentences and select the top-$k$ as the extracted summary.

### 5.2.4 Sentence Generation

We also provide the generation for the abstractive summarization. The summary generator in our model uses autoregressive language modeling, which is defined as estimating the probability distribution of an existing token given its previous tokens in an input sequence. As similiar as the previous works like (Fabbri et al., 2019; Li et al., 2020a), we develop and evaluate our model on autoregressive language modeling.

### 5.2.5 Extension to Multi-Document

Our framework can establish the document-level relationship in the same way as the sentence-level, by just adding *document* nodes for multiple documents (i.e., adding the `[CLS]` token in front of each document) and calculate the *document*↔*sentence* (*d2s*, *s2d*), *document*↔*token* (*d2t*, *t2d*) and *document-to-document* (*d2d*) attention patterns. Therefore, it can be easily adapted from the single-document to multi-document summarization.

### 5.2.6 Discussions

The most relevant approaches to this work are Longformer (Beltagy et al., 2020) and ETC (Ravula et al., 2020) which use a hierarchical attention pattern to scale Transformers to long documents. Compared to these two methods, we formulate the Transformer as multi-

granularity graph attention patterns, which can better encode heterogeneous node types and different edge connections. More specifically, Longformer treats the input sequence as one sentence with the single tokens marked as global. In contrast, we consider the input sequence as multi-sentence units by using *sentence-to-sentence* attention, which is able to capture the inter-sentence relationships in the complex document. Additionally, we introduce *entity-to-entity* attention pattern to facilitate the connection of relevant subjects and preserve global context, which are ignored in both Longformer and ETC. Moreover, our model is more flexible to be extended to the multi-document setting.

## 5.3 Experiments

### 5.3.1 Datasets

**CNN/DailyMail** is the most widely used benchmark dataset for single-document summarization (Zhang et al., 2019a; Jia et al., 2020). The standard dataset split contains 287,227/13,368 /11,490 samples for train/validation/test. To be comparable with other baselines, we follow the data processing in (Liu and Lapata, 2019b; See et al., 2017).

**Multi-News** is a large-scale dataset for multi-document summarization introduced in (Fabbri et al., 2019), where each sample is composed of 2-10 documents and a corresponding human-written summary. Following (Fabbri et al., 2019), we split the dataset into 44,972/5,622/5,622 for train/validation/test. The average length of input documents and output summaries are 2,103.5 tokens and 263.7 tokens, respectively. Given the N input documents, we truncate each input document to the first L/N tokens. Then we concatenate the truncated input documents into a sequence followed by their original order. Due to the memory limitation, we truncate

input length L to 1,024 tokens. But if the memory capacity allows, our model can process the max input length = 4,096.

Since the dataset only contains abstractive gold summaries, it is not readily suited to training models, which needs the binary label for each sentence. So we follow the previous work of (Zhou et al., 2018) on constructing the extractive summary labels, which produces the gold-label sequences by greedily optimizing R-2 score with the gold-standard summary.

### 5.3.2 Baselines and Metrics

**Models: BERT** (or RoBERTa) (Devlin et al., 2018; Liu et al., 2019b) is a Transformer-based model for text understanding through masking language models. **HIBERT** (Zhang et al., 2019a) proposed a hierarchical Transformer model where it first encodes each sentence using the sentence Transformer encoder, and then encoded the whole document using the document Transformer encoder. **HSG**, HDSG (Wang et al., 2020a) formulated the input text as the heterogeneous graph which contains different granularity semantic nodes, (like word, sentence, document nodes) and connected the nodes with the TF-IDF. HSG used CNN and BiLSTM to initialize the node representation and updated the node representation by iteratively passing messages by Graph Attention Network (GAT). In the end, the final sentence nodes representation is used to select the summary sentence. **HAHsum** (Jia et al., 2020) constructed the input text as the heterogeneous graph containing the word, named entity, and sentence node. HAHsum used a pre-trained ALBERT to learn the node initial representation and then adapted GAT to iteratively learn node hidden representations. **MGsum** (Hanqi Jin, 2020) treated documents, sentences, and words as the different granularity of semantic units,

and connected these semantic units within a multi-granularity hierarchical graph. They also proposed a model based on GAT to update the node representation. **ETC** (Narayan et al., 2020), and Longformer (Beltagy et al., 2020) are two pre-trained models to capture hierarchical structures among input documents through the sparse attention mechanism.

**Abstractive Models: Hi-MAP** (Fabbri et al., 2019) expands the pointer-generator network model into a hierarchical network and integrates an MMR module to calculate sentence-level scores. **Graphsum** (Li et al., 2020a) leverage the graph representations of documents by processing input documents as the hierarchical structure with a pre-trained language model to generate the abstractive summary.

We use unigram, bigram, and longest common subsequence of Rouge F1 (denoted as R-1, R-1 and R-L) (Lin and Och, 2004)[1] to evaluate the summarization qualities. Note that the experimental results of baselines are from the original papers.

### 5.3.3 Implementation Detail

Our model HetFormer is initialized using the Longformer pretrained checkpoints `longformer-base-4096`[2], which is further pertained using the standard masked language model task on the Roberta checkpoints `roberta-base`[3] with the documents of max length 4,096. We apply dropout with probability 0.1 before all linear layers in our models. The proposed

---

[1] `https://pypi.org/project/rouge/`

[2] `https://github.com/allenai/longformer`

[3] `https://github.com/huggingface/transformers`

| Model | R-1 | R-2 | R-L |
|---|---|---|---|
| HiBERT (Zhang et al., 2019a) | 42.31 | 19.87 | 38.78 |
| HSG (Wang et al., 2020a) | 42.95 | 19.76 | 39.23 |
| $\text{HAHsum}_{\text{Large}}$ (Jia et al., 2020) * | 44.67 | 21.30 | 40.75 |
| MatchSum (Zhong et al., 2020) | 44.41 | 20.86 | 40.55 |
| $\text{BERT}_{\text{Base}}$ (Devlin et al., 2018) | 41.55 | 19.34 | 37.80 |
| $\text{RoBERTa}_{\text{Base}}$ (Liu et al., 2019b) | 42.99 | 20.60 | 39.21 |
| $\text{ETC}_{\text{Base}}$ (Narayan et al., 2020) | 43.43 | 20.54 | 39.58 |
| $\text{Longformer}_{\text{Base}}$ (Beltagy et al., 2020) | 43.20 | 20.38 | 39.61 |
| $\text{HETFORMER}_{\text{Base}}$ | 44.55 | 20.82 | 40.37 |
| $\text{HETFORMER}_{\text{Base}}$ Gen | **45.04** | **21.69** | **40.87** |

TABLE XV. Rouge F1 scores on test set of CNN/DailyMail. *Note that $\text{HAHsum}_{\text{Large}}$ uses large verision while the proposed model is based on the base version.

model follows the Longformer-base architecture, where the number of $d_{\text{model}}$ hidden units in our models is set as 768, the $d_h$ hidden size is 64, the layer number is 12 and the number of heads is 12. We train our model for 500K steps on the TitanRTX, 24G GPU with gradient accumulation in every two steps with Adam optimizers. Learning rate schedule follows the strategies with warming-up on first 10,000 steps (Vaswani et al., 2017). We select the top-3 checkpoints according to the evaluation loss on validation set and report the averaged results on the test set.

For the testing stage, we select top-3 sentences for CNN/DailyMail and top-9 for Multi-News according to the average length of their human-written summaries. Trigram blocking is used to reduce repetitions.

| Model | R-1 | R-2 | R-L |
|---|---|---|---|
| HiBERT (Zhang et al., 2019a) | 44.32 | 15.11 | 29.26 |
| Hi-MAP (Fabbri et al., 2019) | 45.21 | 16.29 | 41.39 |
| HDSG (Wang et al., 2020a) | 46.05 | 16.35 | 42.08 |
| MatchSum (Zhong et al., 2020) | 46.20 | 16.51 | 41.89 |
| MGsum$_{Base}$ (Hanqi Jin, 2020) | 45.04 | 15.98 | - |
| Graphsum$_{Base}$ (Li et al., 2020a) | 46.07 | 17.42 | - |
| Longformer$_{Base}$ (Beltagy et al., 2020) | 45.34 | 16.00 | 40.54 |
| HetFormer$_{Base}$ | 46.21 | 17.49 | 42.43 |
| HetFormer$_{\text{Base}}$ Gen | **46.41** | **17.71** | **42.57** |

TABLE XVI. Rouge F1 scores on test set of Multi-News. '-' means that the original paper did not report the result.

### 5.3.4 Summerization Results

As shown in Table XV, our approach outperforms or is on par with current state-of-the-art baselines. Longformer and ETC outperforms the hierarchical structure model using fully-connected attention model HiBERT, which shows the supreme of using sparse attention by capturing more relations (e.g., token-to-sentence and sentence-to-token). Comparing to the pre-trained models using sparse attention, HetFormer considering the heterogeneous graph structure among the text input outperforms Longformer and ETC. Moreover, HetFormer achieves competitive performance compared with GNN-based models, such as HSG and HAHsum. Our model is slightly lower than the performance of HAHsum$_{\text{large}}$. But it uses large architecture (24 layers with about 400M parameters), while our model builds on the base model (12 layers with about 170M parameters). Table XVI shows the results of multi-document summarization. Our model outperforms all the extractive and abstractive baselines. These results reveal the importance of modeling the longer document to avoid serious information loss.

| | BERT | RoBERTa | Longformer | Ours |
|---|---|---|---|---|
| Memory Cost | 3,057M | 3,540M | 1,650M | 1,979M |

TABLE XVII. Memory cost of different pre-trained models

### 5.3.5 Memory Cost

Compared with the self-attention component requiring quadratic memory complexity in original Transformers, the proposed model only calculates the position where attention pattern mask=1, which can significantly save the memory cost. To verify that, we show the memory costs of BERT, RoBERTa, Longformer and HetFormer base-version model on the CNN/DailyMail dataset with the same configuration (input length = 512, batch size = 1).

From the results in Table XVII, we can see that HetFormer only takes 55.9% memory cost of RoBERTa model and also does not take too much more memory than Longformer.

### 5.3.6 Ablation Study

To verify the importance of the design choices of the attention patterns, we attempted different variants and reported their experiment results. In order to make the ablation study more manageable, we train each configuration for 500K steps on the single-document CNN/DailyMail dataset, then report the Rouge score on the test set.

The top of Table XVIII shows the influence of different ways of configuring the window sizes per layer. We find that enlarging the window size from the bottom to the top layer results in the best performance (from 32 to 512). But the reverse way leads to worse performance (from 512

to 32). And using a fixed window size (the average of window sizes of the other configuration) leads to a middle-level performance.

The middle of Table XVIII presents the impact of incorporating the sentence node in the attention pattern. In implementation, no sentence node means that we delete the `[CLS]` tokens of the document input and use the average representation of each token in the sentences as the sentence representation. We observe that without using the sentence node to fully connect with the other tokens could decrease the performance.

The bottom of Table XVIII shows the influence of using the entity node. We can see that without the entity node, the performance will decrease. It demonstrates that facilitating the connection of relevant subjects can preserve the global context, which can benefit the summarization task.

| Model | R-1 | R-2 | R-L |
|---|---|---|---|
| Decreasing w (from 512 to 32) | 43.98 | 20.33 | 39.39 |
| Fixed w (=128) | 43.92 | 20.43 | 39.43 |
| Increasing w (from 32 to 512) | **44.55** | **20.82** | **40.37** |
| No Sentence node | 42.15 | 20.12 | 38.91 |
| No Entity node | 43.65 | 20.40 | 39.28 |

TABLE XVIII. Top: changing window size across layers. Middle: entity-to-entity attention pattern influence. Bottom: sentence-to-sentence attention pattern influence

## 5.4 Background

### 5.4.1 Graph-enhanced Summarization

In the recent state-of-the-art summarization models, there is a trend to extract the structure from the text to formulate the document text as a hierarchical structure or heterogeneous graph (Liu et al., 2020). HiBERT (Zhang et al., 2019a), GraphSum (Li et al., 2020a) and HT (Liu and Lapata, 2019a) consider the word-level, sentence-level and document-level of the input text to formulate the hierarchical structure. MGSum (Hanqi Jin, 2020), ASGARD (Huang et al., 2020), HSG (Wang et al., 2020a) and HAHSum (Jia et al., 2020) construct the source article as a heterogeneous graph where words, sentences, and entities are used as the semantic nodes and they iteratively update the sentence nodes representation which is used to do the sentence extraction.

The limitation of those models is that they use pre-trained methods as the feature-based model to learn the node feature and build GNN layers upon the node which brings more training parameters than just using pre-trained methods. Compared with those models, our work can achieve the same thing but using the lite framework. Moreover, these models typically limit inputs to $n = 512$ tokens since the $O(n^2)$ cost of attention. Due to the long source article, when applying BERT or RoBERTa to the summarization task, they need to truncate source documents into one or several smaller block input (Li et al., 2020a; Jia et al., 2020; Huang et al., 2020).

### 5.4.2 Structure Transformer

(Huang et al., 2021) proposed an efficient encoder-decoder attention with head-wise positional strides, which yields ten times faster than existing full attention models and can be scale to long documents. (Liu et al., 2021) leveraged the syntactic and semantic structures of text to improve the Transformer and achieved nine times speedup. Our model focuses on the different direction to use graph-structured sparse attention to capture the long term dependence on the long text input. The most related approaches to the work presented in this paper are Longformer (Beltagy et al., 2020) and ETC (Ravula et al., 2020) which feature a very similar global-local attention mechanism and take advantage of the pre-trained model RoBERTa. Except Longformer has a single input sequence with some tokens marked as global (the only ones that use full attention), while the global tokens in the ETC is pre-trained with CPC loss. Comparing with those two works, we formulate the heterogeneous attention mechanism, which can consider the word-to-word, word-to-sen, sen-to-word and entity-to-entity attention.

### 5.4.3 Graph Transformer

With the great similarity between the attention mechanism used in both Transformer (Vaswani et al., 2017) and Graph Attention network (Veličković et al., 2017), there are several recent Graph Transformer works recently. Such as GTN (Yun et al., 2019), HGT (Hu et al., 2020), (Fan et al., 2021) and HetGT (Yao et al., 2020) formulate the different type of the attention mechanisms to capture the node relationship in the graph.

The major difference between of our work and Graph Transformer is that the input of graph transformer is structural input, such as graph or dependence tree, but the input of

our HeterFormer is unstructured text information. Our work is to convert the transformer to structural structure so that it can capture the latent relation in the unstructured text, such as the word-to-word, word-to-sent, sent-to-word, sent-to-sent and entity-to-entity relations.

# CHAPTER 6

# CONCLUSION

(Part of the chapter was previously published in (Liu et al., 2019; Liu et al., 2020; Liu et al., 2021; Liu et al., 2021)).

To summarize, this dissertation aims to answer two questions that commonly appear in knowledge-enhanced text generation: how to acquire knowledge (knowledge acquisition) and how to incorporate knowledge to facilitate text generation (knowledge incorporation). In terms of knowledge acquisition, the main content of our dissertation is divided into two sections according to different sources of knowledge enhancement, namely internal knowledge enhancement and external knowledge enhancement. In terms of knowledge incorporation, we discuss four specific ideas and technical solutions that incorporate the knowledge to enhance the text generation models in each section.

The main contributions of our works are summarized as follows:

- Question refinement aims to refine ill-formed questions, which typically includes various types of subtasks such as spelling error correction, background removal and word order refinement. Instead of tackle these subtasks separately, we develop a unified model, based on Seq2Seq, to handle this task in a data-driven way. We improve the question representation by incorporating character embedding and contextual word embedding such as BERT. To make the refinement process more controllable, we combine Seq2Seq

model with deep reinforcement learning. We define a sequence generator by optimizing for a combination of imposed reward functions. The experimental results show that our method can not only produce more readable question but also significantly improves the retrieval ability of question for downstream QA system.

Question refinement is a challenging task and there are several directions to improve. One direction is to develop the advanced method, such as creating different awards that are more suitable to deal with the three subtasks. Besides, In our setting, the ill-formed and well-formed questions still need to be paired. In most of realistic cases, we only have a pool of well-formed. We seek to use inverse reinforcement learning (Wang et al., 2018) to learn the intrinsic representation of the well-formed question. Therefore, given an ill-formed question, the model can refine it to the well-formed. Finally, it is also interesting to make use of the result of question refinement to improve other related tasks such as question understanding (Braun et al., 2017) and question recommendation (San Pedro and Karatzoglou, 2014).

- We have presented a KG-augmented approach KG-BART based on pre-trained BART for generative commonsense reasoning. Through capturing the relations among concepts over a KG, KG-BART can generate high-quality sentences even in the unseen concept sets. KG-BART further considers the neighbor entities of each concept node as to generate more natural and logical sentences. It can also be extended to any seq2seq pre-trained language generation models, like T5 (Raffel et al., 2020) and MASS (Song et al., 2019).

Experimental results demonstrate that KG-BART has better abilities of both commonsense reasoning and text generalization.

- We have proposed a novel syntactic and semantic structure-aware non-autoregressive Transformer model **SNAT** for NMT. The proposed model aims at reducing the computational cost in inference as well as keeping the quality of translation by incorporating both syntactic and semantic structures existing among natural languages into a non-autoregressive Transformer. In addition, we have also designed an intermediate latent alignment regularization within target sentences to better learn the long-term token dependencies. Comprehensive experiments and analysis on two real-world datasets (i.e., WMT14 En→De and WMT16 En→Ro) verify the efficiency and effectiveness of our proposed approach.

- For the task of long-text extractive summarization, we have proposed HetFormer, using multi-granularity sparse attention to represent the heterogeneous graph among texts.Experiments show that the proposed model can achieve comparable performance on a single-document summarization task, as well as state-of-the-art performance on the multi-document summarization task with longer input document. In our future work, we plan to expand the edge from the binary type (connect or disconnect) to more plentiful semantic types, i.e., *is-a*, *part-of*, and others (Zhang et al., 2020b).

# APPENDICES

## .1 ACM Copyright Letter

"Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

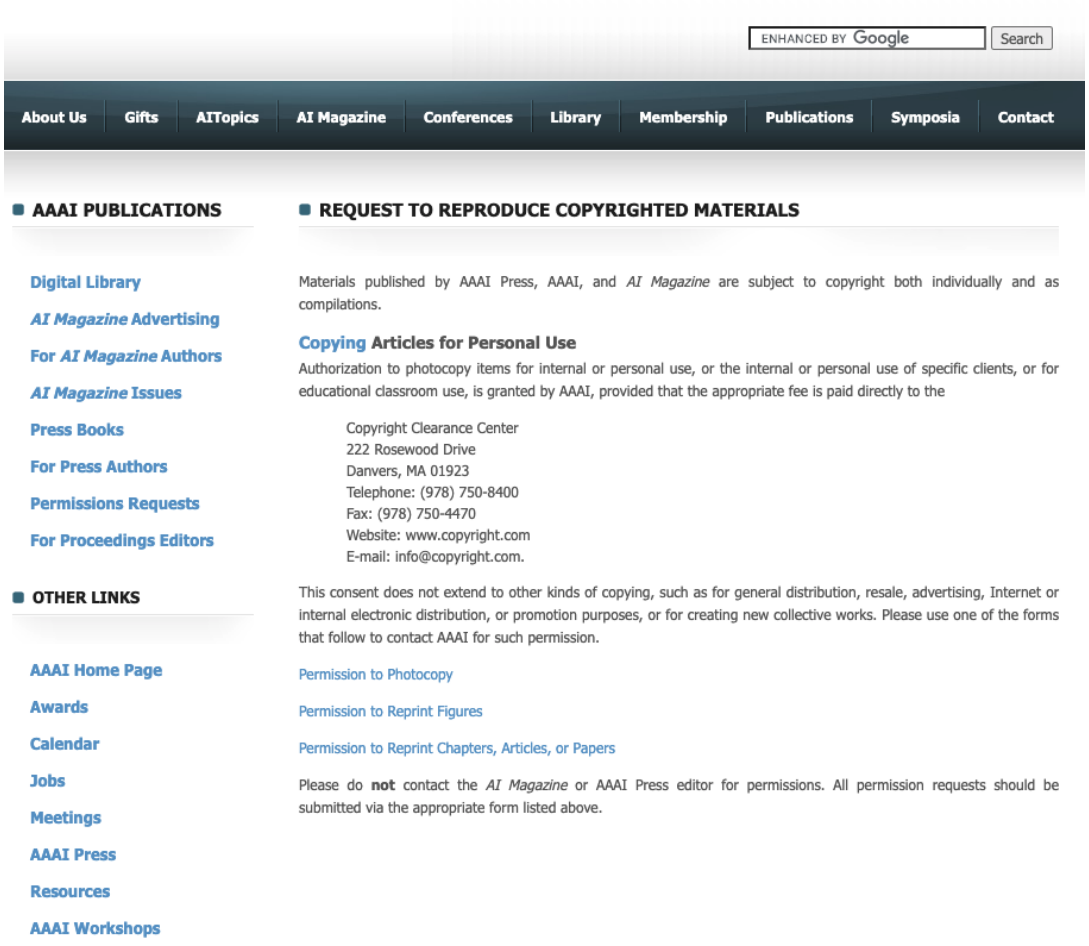
Contributing complete papers to any edited collection of reprints for which the author is not the editor, requires permission and usually a republication fee.

Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected)." [1]

---

[1] http://authors.acm.org/main.html

## .2 AAAI Copyright Letter



ENHANCED BY Google Search

About Us | Gifts | AITopics | AI Magazine | Conferences | Library | Membership | Publications | Symposia | Contact

**AAAI PUBLICATIONS**

- Digital Library
- *AI Magazine* Advertising
- For *AI Magazine* Authors
- *AI Magazine* Issues
- Press Books
- For Press Authors
- Permissions Requests
- For Proceedings Editors

**OTHER LINKS**

- AAAI Home Page
- Awards
- Calendar
- Jobs
- Meetings
- AAAI Press
- Resources
- AAAI Workshops

**REQUEST TO REPRODUCE COPYRIGHTED MATERIALS**

Materials published by AAAI Press, AAAI, and *AI Magazine* are subject to copyright both individually and as compilations.

**Copying Articles for Personal Use**

Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, or for educational classroom use, is granted by AAAI, provided that the appropriate fee is paid directly to the

Copyright Clearance Center
222 Rosewood Drive
Danvers, MA 01923
Telephone: (978) 750-8400
Fax: (978) 750-4470
Website: www.copyright.com
E-mail: info@copyright.com.

This consent does not extend to other kinds of copying, such as for general distribution, resale, advertising, Internet or internal electronic distribution, or promotion purposes, or for creating new collective works. Please use one of the forms that follow to contact AAAI for such permission.

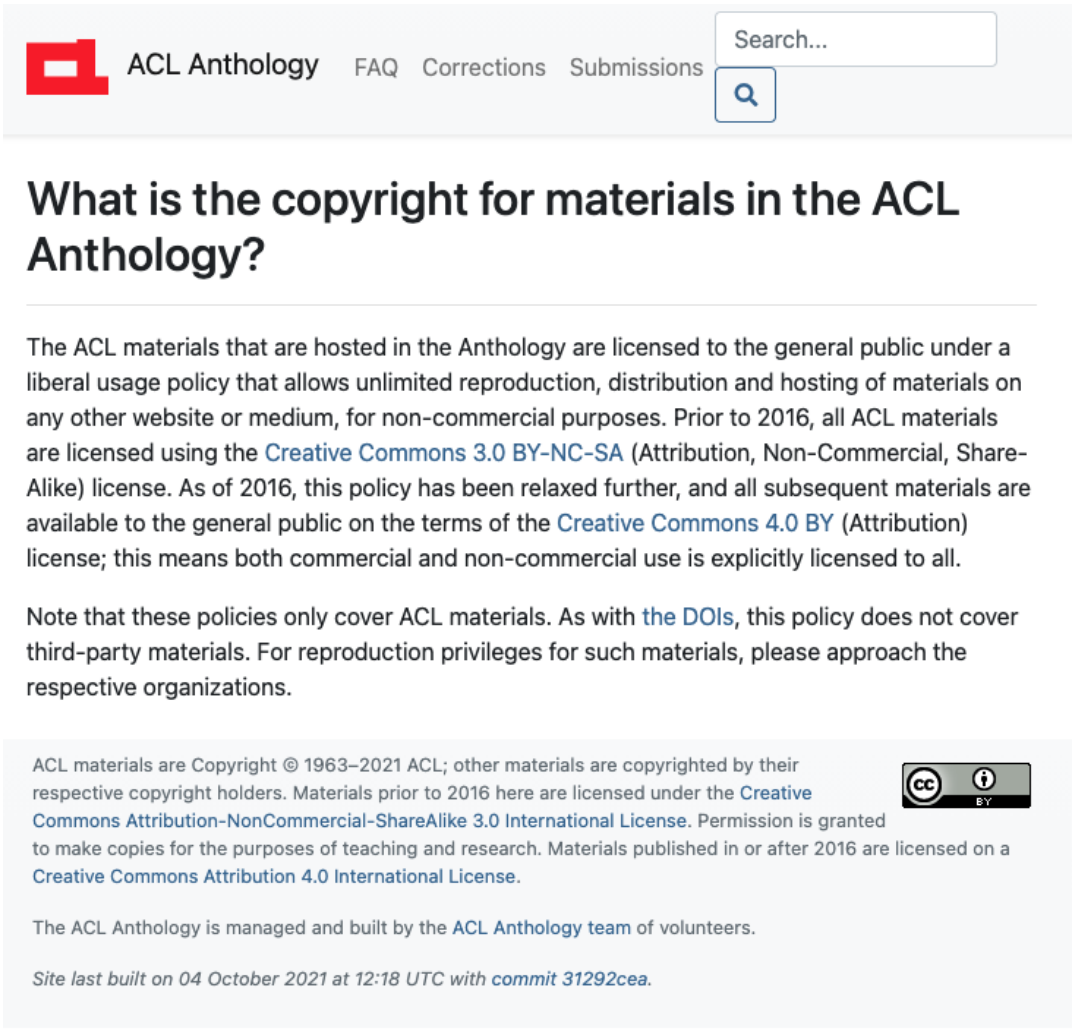
Permission to Photocopy

Permission to Reprint Figures

Permission to Reprint Chapters, Articles, or Papers

Please do **not** contact the *AI Magazine* or AAAI Press editor for permissions. All permission requests should be submitted via the appropriate form listed above.

## .3 ACL Copyright letter



ACL Anthology FAQ Corrections Submissions Search...

# What is the copyright for materials in the ACL Anthology?

The ACL materials that are hosted in the Anthology are licensed to the general public under a liberal usage policy that allows unlimited reproduction, distribution and hosting of materials on any other website or medium, for non-commercial purposes. Prior to 2016, all ACL materials are licensed using the Creative Commons 3.0 BY-NC-SA (Attribution, Non-Commercial, Share-Alike) license. As of 2016, this policy has been relaxed further, and all subsequent materials are available to the general public on the terms of the Creative Commons 4.0 BY (Attribution) license; this means both commercial and non-commercial use is explicitly licensed to all.

Note that these policies only cover ACL materials. As with the DOIs, this policy does not cover third-party materials. For reproduction privileges for such materials, please approach the respective organizations.

ACL materials are Copyright © 1963–2021 ACL; other materials are copyrighted by their respective copyright holders. Materials prior to 2016 here are licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 International License. Permission is granted to make copies for the purposes of teaching and research. Materials published in or after 2016 are licensed on a Creative Commons Attribution 4.0 International License.

The ACL Anthology is managed and built by the ACL Anthology team of volunteers.

*Site last built on 04 October 2021 at 12:18 UTC with commit 31292cea.*

## .4 arXiv.org Copyright Letter

# arXiv.org - Non-exclusive license to distribute

The URI http://arxiv.org/licenses/nonexclusive-distrib/1.0/ is used to record the fact that the submitter granted the following license to arXiv.org on submission of an article:

- I grant arXiv.org a perpetual, non-exclusive license to distribute this article.
- I certify that I have the right to grant this license.
- I understand that submissions cannot be completely removed once accepted.
- I understand that arXiv.org reserves the right to reclassify or reject any submission.

### Revision history

2004-01-16 - License above introduced as part of arXiv submission process
2007-06-21 - This HTML page created

*Contact*

## CITED LITERATURE


Anderson, P., Fernando, B., Johnson, M., and Gould, S.: Spice: Semantic propositional image caption evaluation. In Proceedings of ECCV, pages 382–398. Springer, 2016.

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., and Schmidt, L.: Practical and optimal lsh for angular distance. In Proceedings of the Conference of Neural Information Processing Systems, pages 1225–1233, 2015.

Angeli, G., Premkumar, M. J. J., and Manning, C. D.: Leveraging linguistic structure for open domain information extraction. In Proceedings of the Conference of Association for Computational Linguistics, pages 344–354, 2015.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z.: Dbpedia: A nucleus for a web of open data. In The semantic web, pages 722–735. Springer, 2007.

Ba, J. L., Kiros, J. R., and Hinton, G. E.: Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., Bengio, Yoshua Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y.: An actor-critic algorithm for sequence prediction. In 5th International Conference on Learning Representations, ICLR 2017, 2017.

Bahdanau, D., Cho, K., and Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

Banerjee, S. and Lavie, A.: Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In Proceedings of the ACL workshop, pages 65–72, 2005.

Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Piao, S., Gao, J., Zhou, M., et al.: Unilmv2: Pseudo-masked language models for unified language model pre-training. arXiv preprint arXiv:2002.12804, 2020.

Bao, Y., Zhou, H., Feng, J., Wang, M., Huang, S., Chen, J., and Li, L.: Pnat: Non-autoregressive transformer by position learning. 2019.

Barto, A. G. and Sutton, R. S.: Reinforcement learning. Neural systems for control, pages 7–29, 1998.

Beltagy, I., Peters, M. E., and Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150, 2020.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of SIGMOD, pages 1247–1250, 2008.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In Proceedings of NeurIPS, pages 2787–2795, 2013.

Bordino, I., Castillo, C., Donato, D., and Gionis, A.: Query similarity by projecting the query-flow graph. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pages 515–522. ACM, 2010.

Braun, D., Hernandez-Mendez, A., Matthes, F., and Langen, M.: Evaluating natural language understanding services for conversational question answering systems. In Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, pages 174–185, 2017.

Britz, D., Goldie, A., Luong, M.-T., and Le, Q.: Massive exploration of neural machine translation architectures. arXiv preprint arXiv:1703.03906, 2017.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. In Proceedings of NeurIPS, 2020.

Buck, C., Bulian, J., Ciaramita, M., Gajewski, W., Gesmundo, A., Houlsby, N., and Wang, W.: Ask the right questions: Active question reformulation with reinforcement learning. ICLR, 2018.

Cai, D. and Lam, W.: Graph transformer for graph-to-sequence learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 7464–7471, 2020.

Carbonell, J. and Goldstein, J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries. In Proceedings of the SIGIR, pages 335–336, 1998.

Carpineto, C. and Romano, G.: A survey of automatic query expansion in information retrieval. CSUR, 44(1):1, 2012.

Chan, W., Kitaev, N., Guu, K., Stern, M., and Uszkoreit, J.: Kermit: Generative insertion-based modeling for sequences. arXiv preprint arXiv:1906.01604, 2019.

Chen, Q., Li, M., and Zhou, M.: Improving query spelling correction using web search results. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007.

Child, R., Gray, S., Radford, A., and Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509, 2019.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

Choi, E., Hewlett, D., Uszkoreit, J., Polosukhin, I., Lacoste, A., and Berant, J.: Coarse-to-fine question answering for long documents. In ACL, volume 1, pages 209–220, 2017.

Dahab, M. Y., Alnofaie, S., and Kamel, M.: A tutorial on information retrieval using query expansion. In Intelligent Natural Language Processing: Trends and Applications, pages 761–776. Springer, 2018.

Dai, A. M. and Le, Q. V.: Semi-supervised sequence learning. In Proceedings of the Conference of Neural Information Processing Systems, pages 3079–3087, 2015.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. In Proceedings of the Conference of Association for Computational Linguistics, pages 2978–2988, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the North American Chapter of ssociation for Computational Linguistics, pages 4171–4186, 2018.

Dong, L., Mallinson, J., Reddy, S., and Lapata, M.: Learning to paraphrase for question answering. arXiv preprint arXiv:1708.06022, 2017.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W.: Unified language model pre-training for natural language understanding and generation. In Proceedings of NeurIPS, pages 13063–13075, 2019.

Erkan, G. and Radev, D. R.: Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of artificial intelligence research, 22:457–479, 2004.

Fabbri, A. R., Li, I., She, T., Li, S., and Radev, D. R.: Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In Proceedings of the Conference of Association for Computational Linguistics, pages 1074–1084, 2019.

Fan, Z., Liu, Z., Zhang, J., Xiong, Y., Zheng, L., and Yu, P. S.: Continuous-time sequential recommendation with temporal graph collaborative transformer. In Proceedings of ACM International Conference on Information and Knowledge Management, 2021.

Faruqui, M. and Das, D.: Identifying well-formed natural language questions. arXiv preprint arXiv:1808.09419, 2018.

Feng, M., Xiang, B., Glass, M. R., Wang, L., and Zhou, B.: Applying deep learning to answer selection: A study and an open task. In Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on, pages 813–820. IEEE, 2015.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N.: Convolutional sequence to sequence learning. In International Conference on Machine Learning, pages 1243–1252. PMLR, 2017.

Gehrmann, S., Deng, Y., and Rush, A. M.: Bottom-up abstractive summarization. In Proceedings of the Conference of Neural Information Processing Systems, pages 4098–4109, 2018.

Ghazvininejad, M., Karpukhin, V., Zettlemoyer, L., and Levy, O.: Aligned cross entropy for non-autoregressive machine translation. In Proceedings of the International Conference on Machine Learning, pages 9330–9338, 2020.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L.: Mask-predict: Parallel decoding of conditional masked language models. 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 6114–6123, 2019.

Goodfellow, I., Bengio, Y., and Courville, A.: Deep learning. MIT press, 2016.

Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

Greensmith, E., Bartlett, P. L., and Baxter, J.: Variance reduction techniques for gradient estimates in reinforcement learning. Journal of Machine Learning Research, 5(Nov):1471–1530, 2004.

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R.: Non-autoregressive neural machine translation. In International Conference on Learning Representations, 2018.

Gu, J., Lu, Z., Li, H., and Li, V. O.: Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393, 2016.

Guan, J., Wang, Y., and Huang, M.: Story ending generation with incremental encoding and commonsense knowledge. In Proceedings of AAAI, volume 33, pages 6473–6480, 2019.

Guo, J., Xu, G., Li, H., and Cheng, X.: A unified and discriminative model for query refinement. In SIGIR, pages 379–386. ACM, 2008.

Guo, J., Tan, X., He, D., Qin, T., Xu, L., and Liu, T.-Y.: Non-autoregressive neural machine translation with enhanced decoder input. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3723–3730, 2019.

Hanqi Jin, Tianming Wang, X. W.: Multi-granularity interaction network for extractive and abstractive multi-document summarization. In Proceedings of the Conference of Association for Computational Linguistics, pages 6244–6254, 2020.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., and Ma, W.-Y.: Dual learning for machine translation. In NIPS, pages 820–828, 2016.

Hendrycks, D. and Gimpel, K.: Bridging nonlinearities and stochastic regularizers with gaussian error linear units. 2016.

Hendrycks, D. and Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P.: Teaching machines to read and comprehend. In Proceedings of the Conference of Neural Information Processing Systems, pages 1693–1701, 2015.

Hoang, H. and Koehn, P.: Design of the moses decoder for statistical machine translation. In Software Engineering, Testing, and Quality Assurance for Natural Language Processing, pages 58–65, 2008.

Hochreiter, S. and Schmidhuber, J.: Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y.: The curious case of neural text degeneration. 2020.

Horibe, F.: Managing knowledge workers: New skills and attitudes to unlock the intellectual capital in your organization. John Wiley & Sons, 1999.

Howard, J. and Ruder, S.: Universal language model fine-tuning for text classification. In Proceedings of the Conference of Association for Computational Linguistics, page 328–339, 2018.

Hu, Z., Dong, Y., Wang, K., and Sun, Y.: Heterogeneous graph transformer. In Proceedings of the Web Conference, pages 2704–2710, 2020.

Huang, L., Bras, R. L., Bhagavatula, C., and Choi, Y.: Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. In Proceedings of EMNLP, pages 2391–2401, 2019.

Huang, L., Cao, S., Parulian, N., Ji, H., and Wang, L.: Efficient attentions for long document summarization. In Proceedings of the North American Chapter of the Association for Computational Linguistics, 2021.

Huang, L., Wu, L., and Wang, L.: Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In Proceedings of the Conference of Association for Computational Linguistics, page 5094–5107, 2020.

Jaques, N., Gu, S., Turner, R. E., and Eck, D.: Tuning recurrent neural networks with reinforcement learning. arXiv preprint arXiv:1611.02796, 2016.

Jaques, N., Gu, S., Turner, R. E., and Eck, D.: Tuning recurrent neural networks with reinforcement learning. 2017.

Jia, R., Cao, Y., Tang, H., Fang, F., Cao, C., and Wang, S.: Neural extractive summarization with hierarchical attentive heterogeneous graph network. In Proceedings of the Conference of Neural Information Processing Systems, pages 3622–3631, 2020.

Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., and Shazeer, N.: Fast decoding in sequence models using discrete latent variables. In Proceedings of the 35th International Conference on Machine Learning, eds. J. Dy and A. Krause, volume 80 of Proceedings of Machine Learning Research, pages 2390–2399, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F.: Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the ICML, 2020.

Kim, Y.: Convolutional neural networks for sentence classification. In Proceedings of EMNLP, pages 1746–1751, 2014.

Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kingma, D. P. and Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In Advances in neural information processing systems, pages 10215–10224, 2018.

Kitaev, N., Kaiser, Ł., and Levskaya, A.: Reformer: The efficient transformer. In Proceedings of the International Conference on Learning Representations, 2020.

Klusch, M., Kapahnke, P., Schulte, S., Lecue, F., and Bernstein, A.: Semantic web service search: A brief survey. KI-Künstliche Intelligenz, 30(2):139–147, 2016.

Kneser, R. and Ney, H.: Improved backing-off for m-gram language modeling. In icassp, volume 1, page 181e4, 1995.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. In Proceedings of the International Conference on Learning Representations, 2020.

LeCun, Y., Bengio, Y., and Hinton, G.: Deep learning. nature, 521(7553):436–444, 2015.

Lee, J., Mansimov, E., and Cho, K.: Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Proceedings of the 2018 Conference on Empirical Methods

in Natural Language Processing, pages 1173–1182, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of ACL, pages 7871–7880, 2020.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D.: Deep reinforcement learning for dialogue generation. arXiv preprint arXiv:1606.01541, 2016.

Li, W., Xiao, X., Liu, J., Wu, H., Wang, H., and Du, J.: Leveraging graph to improve abstractive multi-document summarization. In Proceedings of the Conference of Association for Computational Linguistics, pages 6232—6243, 2020.

Li, X., Meng, Y., Yuan, A., Wu, F., and Li, J.: Lava nat: A non-autoregressive translation model with look-around decoding and vocabulary attention. arXiv preprint arXiv:2002.03084, 2020.

Li, Y.: Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274, 2017.

Li, Z., Lin, Z., He, D., Tian, F., Qin, T., Wang, L., and Liu, T.-Y.: Hint-based training for non-autoregressive machine translation. In Proceedings of the International Conference on Learning Representations, 2020.

Li, Z., Jiang, X., Shang, L., and Li, H.: Paraphrase generation with deep reinforcement learning. arXiv preprint arXiv:1711.00279, 2017.

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N.: Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. arXiv preprint arXiv:1611.00020, 2016.

Libovický, J. and Helcl, J.: End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3016–3021, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Lin, B. Y., Chen, X., Chen, J., and Ren, X.: Kagnet: Knowledge-aware graph networks for commonsense reasoning. In Proceedings of EMNLP, pages 2829–2839, 2019.

Lin, B. Y., Shen, M., Zhou, W., Zhou, P., Bhagavatula, C., Choi, Y., and Ren, X.: Commongen: A constrained text generation challenge for generative commonsense reasoning. In Proceedings of EMNLP findings, 2020.

Lin, C.-Y.: Rouge: A package for automatic evaluation of summaries. In Proceedings of Text summarization branches out, pages 74–81, 2004.

Lin, C.-Y. and Och, F. J.: Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In Proceedings of the Conference of Association for Computational Linguistics, pages 605–612, 2004.

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., and Wang, P.: K-bert: Enabling language representation with knowledge graph. In Proceedings of AAAI, pages 2901–2908, 2020.

Liu, Y. and Lapata, M.: Hierarchical transformers for multi-document summarization. In Proceedings of the Conference of Association for Computational Linguistics, pages 5070–5081, 2019.

Liu, Y. and Lapata, M.: Text summarization with pretrained encoders. In Proceedings of the Conference of Neural Information Processing Systems, pages 3730–3740, 2019.

Liu, Y., Wan, Y., He, L., Peng, H., and Yu, P. S.: Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. In Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

Liu, Y., Wan, Y., Zhang, J.-G., Zhao, W., and Yu, P. S.: Enriching non-autoregressive transformer with syntactic and semanticstructures for neural machine translation. In Proceedings of the European Chapter of the Association for Computational Linguistics, 2021.

Liu, Y., Yang, T., You, Z., Fan, W., and Yu, P. S.: Commonsense evidence generation and injection in reading comprehension. In Proceedings of SIGDIAL, pages 61–73, 2020.

Liu, Y., Zhang, C., Yan, X., Chang, Y., and Yu, P. S.: Generative question refinement with deep reinforcement learning in retrieval-based qa system. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pages 1643–1652, 2019.

Liu, Y., Zhang, J.-G., Wan, Y., Xia, C., He, L., and Yu, P. S.: Hetformer: Heterogeneous transformer with sparse attention for long-text extractive summarization. In Proceedings of the Empirical Methods in Natural Language Processing, 2021.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

Loshchilov, I. and Hutter, F.: Decoupled weight decay regularization. In Proceedings of ICLR, 2019.

Lu, J., Yang, J., Batra, D., and Parikh, D.: Neural baby talk. In Proceedings of CVPR, pages 7219–7228, 2018.

Luong, M.-T., Pham, H., and Manning, C. D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

Ma, X., Zhou, C., Li, X., Neubig, G., and Hovy, E.: FlowSeq: Non-autoregressive conditional sequence generation with generative flow. 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 4282–4292, Hong Kong, China, November 2019. Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D.: The stanford corenlp natural language processing toolkit. In Proceedings of the Conference of Association for Computational Linguistics, pages 55–60, 2014.

Marcus, M. P., Santorini, B., Marcinkiewicz, M. A., and Taylor, A.: Treebank-3. Linguistic Data Consortium, Philadelphia, 14, 1999.

Maruf, S., Martins, A. F., and Haffari, G.: Selective attention for context-aware neural machine translation. In Proceedings of the North American Chapter of ssociation for Computational Linguistics, page 3092–3102, 2019.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S.: Recurrent neural network based language model. In Interspeech, volume 2, page 3, 2010.

Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., and Khudanpur, S.: Extensions of recurrent neural network language model. In 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP), pages 5528–5531. IEEE, 2011.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J.: Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

Mondal, A., Dey, M., Das, D., Nagpal, S., and Garda, K.: Chatbot: An automated conversation system for the educational domain. In 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), pages 1–5. IEEE, 2018.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al.: Abstractive text summarization using sequence-to-sequence rnns and beyond. In Proceedings of the CoNLL, pages 280–290, 2016.

Narasimhan, K., Kulkarni, T., and Barzilay, R.: Language understanding for text-based games using deep reinforcement learning. arXiv preprint arXiv:1506.08941, 2015.

Narayan, S., Gardent, C., Cohen, S. B., and Shimorina, A.: Split and rephrase. arXiv preprint arXiv:1707.06971, 2017.

Narayan, S., Maynez, J., Adamek, J., Pighin, D., Bratanič, B., and McDonald, R.: Stepwise extractive summarization and planning with structured transformers. In Proceedings of the Conference of Neural Information Processing Systems, page 4143–4159, 2020.

Nogueira, R., Bulian, J., and Ciaramita, M.: Learning to coordinate multiple reinforcement learning agents for diverse query reformulation. arXiv preprint arXiv:1809.10658, 2018.

Nogueira, R. and Cho, K.: End-to-end goal-driven web navigation. In NIPS, pages 1903–1911, 2016.

Nogueira, R. and Cho, K.: Task-oriented query reformulation with reinforcement learning. In EMNLP, pages 574–583, 2017.

Ooi, J., Ma, X., Qin, H., and Liew, S. C.: A survey of query expansion, query suggestion and query refinement techniques. In ICSECS, pages 112–117. IEEE, 2015.

Oord, A. v. d., Li, Y., and Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

Ott, M., Edunov, S., Grangier, D., and Auli, M.: Scaling neural machine translation. In Proceedings of the Third Conference on Machine Translation: Research Papers, pages 1–9, Brussels, Belgium, October 2018. Association for Computational Linguistics.

Palmer, M., Gildea, D., and Kingsbury, P.: The proposition bank: An annotated corpus of semantic roles. Computational linguistics, 31(1):71–106, 2005.

Pan, B., Li, H., Zhao, Z., Cao, B., Cai, D., and He, X.: Memen: multi-layer embedding with memory networks for machine comprehension. arXiv preprint arXiv:1707.09098, 2017.

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

Pennington, J., Socher, R., and Manning, C. D.: Glove: Global vectors for word representation. In Proceedings of EMNLP, pages 1532–1543, 2014.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L.: Deep contextualized word representations. In Proceedings of the Conference of Association for Computational Linguistics, page 2227–2237, 2018.

Pradhan, S., Moschitti, A., Xue, N., Ng, H. T., Björkelund, A., Uryupina, O., Zhang, Y., and Zhong, Z.: Towards robust linguistic analysis using ontonotes. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, pages 143–152, 2013.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J.: Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR, 2020.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W.: Sequence level training with recurrent neural networks. ICLR, 2016.

Ravula, A., Alberti, C., Ainslie, J., Yang, L., Pham, P. M., Wang, Q., Ontanon, S., Sanghai, S. K., Cvicek, V. o. A. f. C. L., and Fisher, Z.: Etc: Encoding long and structured

inputs in transformers. In Proceedings of the Conference of Neural Information Processing Systems, pages 268–284, 2020.

Rush, A. M., Chopra, S., and Weston, J.: A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685, 2015.

Saharia, C., Chan, W., Saxena, S., and Norouzi, M.: Non-autoregressive machine translation with latent alignments. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1098–1108, Online, November 2020. Association for Computational Linguistics.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y.: Winogrande: An adversarial winograd schema challenge at scale. In Proceedings of AAAI, pages 8732–8734, 2020.

San Pedro, J. and Karatzoglou, A.: Question recommendation for collaborative question answering systems with rankslda. In Proceedings of the 8th ACM Conference on Recommender systems, pages 193–200. ACM, 2014.

Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., and Choi, Y.: Atomic: An atlas of machine commonsense for if-then reasoning. In Proceedings of AAAI, volume 33, pages 3027–3035, 2019.

Sap, M., Rashkin, H., Chen, D., LeBras, R., and Choi, Y.: Socialiqa: Commonsense reasoning about social interactions. arXiv preprint arXiv:1904.09728, 2019.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks, 20(1):61–80, 2008.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P.: Trust region policy optimization. In International Conference on Machine Learning, pages 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

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

Sennrich, R., Haddow, B., and Birch, A.: Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for

Computational Linguistics, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

Seo, M., Min, S., Farhadi, A., and Hajishirzi, H.: Query-reduction networks for question answering. arXiv preprint arXiv:1606.04582, 2016.

Shao, C., Zhang, J., Feng, Y., Meng, F., and Zhou, J.: Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 198–205, 2020.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y.: Minimum risk training for neural machine translation. arXiv preprint arXiv:1512.02433, 2015.

Shu, R., Lee, J., Nakayama, H., and Cho, K.: Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In AAAI, pages 8846–8853, 2020.

Siddharthan, A.: A survey of research on text simplification. ITL-International Journal of Applied Linguistics, 165(2):259–298, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y.: Mass: Masked sequence to sequence pre-training for language generation. In Proceedings of ICML, pages 5926–5936, 2019.

Speer, R., Chin, J., and Havasi, C.: Conceptnet 5.5: an open multilingual graph of general knowledge. In Proceedings of AAAI, pages 4444–4451, 2017.

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J.: Insertion transformer: Flexible sequence generation via insertion operations. In Proceedings of the 36th International Conference on Machine Learning, volume 97, pages 5976–5985, 09–15 Jun 2019.

Sun, X., Gao, J., Micol, D., and Quirk, C.: Learning phrase-based spelling error models from clickthrough data. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 266–274. ACL, 2010.

Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H., and Wu, H.: Ernie: Enhanced representation through knowledge integration. arXiv preprint arXiv:1904.09223, 2019.

Sun, Z., Li, Z., Wang, H., He, D., Lin, Z., and Deng, Z.: Fast structured decoding for sequence models. In Advances in Neural Information Processing Systems, pages 3011–3020, 2019.

Sutskever, I., Vinyals, O., and Le, Q. V.: Sequence to sequence learning with neural networks. In NIPS, pages 3104–3112, 2014.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In NIPS, pages 1057–1063, 2000.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z.: Rethinking the inception architecture for computer vision. In Proceedings of CVPR, pages 2818–2826, 2016.

Talmor, A., Herzig, J., Lourie, N., and Berant, J.: Commonsenseqa: A question answering challenge targeting commonsense knowledge. In Proceedings of NAACL, pages 4149–4158, 2019.

Tan, M., Santos, C. d., Xiang, B., and Zhou, B.: Lstm-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108, 2015.

Tuan, Y.-L., Zhang, J., Li, Y., and Lee, H.-y.: Proximal policy optimization and its dynamic version for sequence generation. arXiv preprint arXiv:1808.07982, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I.: Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D.: Cider: Consensus-based image description evaluation. In Proceedings of CVPR, pages 4566–4575, 2015.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y.: Graph attention networks. In Proceedings of ICLR, 2017.

Vinyals, O. and Le, Q.: A neural conversational model. arXiv preprint arXiv:1506.05869, 2015.

Vyas, A., Katharopoulos, A., and Fleuret, F.: Fast transformers with clustered attention. In Proceedings of the Conference of Neural Information Processing Systems, volume 33, 2020.

Wang, D., Liu, P., Zheng, Y., Qiu, X., and Huang, X.: Heterogeneous graph neural networks for extractive document summarization. In Proceedings of the Conference of Association for Computational Linguistics, page 6209–6219, 2020.

Wang, X., Gao, T., Zhu, Z., Liu, Z., Li, J., and Tang, J.: Kepler: A unified model for knowledge embedding and pre-trained language representation. TACL, 2020.

Wang, X., Chen, W., Wang, Y.-F., and Wang, W. Y.: No metrics are perfect: Adversarial reward learning for visual storytelling. arXiv preprint arXiv:1804.09160, 2018.

Wang, Y., Tian, F., He, D., Qin, T., Zhai, C., and Liu, T.-Y.: Non-autoregressive machine translation with auxiliary regularization. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 5377–5384, 2019.

Wei, B., Wang, M., Zhou, H., Lin, J., and Sun, X.: Imitation learning for non-autoregressive neural machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1304–1312, Florence, Italy, July 2019. Association for Computational Linguistics.

Wilcoxon, F., Katti, S., and Wilcox, R. A.: Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. Selected tables in mathematical statistics, 1:171–259, 1970.

Williams, R. J. and Peng, J.: Function optimization using connectionist reinforcement learning algorithms. Connection Science, 3(3):241–268, 1991.

Wu, C.-H., Yeh, J.-F., and Lai, Y.-S.: Semantic segment extraction and matching for internet faq retrieval. IEEE transactions on knowledge and data engineering, 18(7):930–940, 2006.

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., and Auli, M.: Pay less attention with lightweight and dynamic convolutions. In Proceedings of the International Conference on Learning Representations, 2019.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y.: Neural language correction with character-based attention. arXiv preprint arXiv:1603.09727, 2016.

Yang, P., Li, L., Luo, F., Liu, T., and Sun, X.: Enhancing topic-to-essay generation with external commonsense knowledge. In Proceedings of ACL, pages 2002–2012, 2019.

Yang, P., Luo, F., Chen, P., Li, L., Yin, Z., He, X., and Sun, X.: Knowledgeable storyteller: A commonsense-driven generative model for visual storytelling. In Proceedings of IJCAI, pages 5356–5362, 2019.

Yao, L., Mao, C., and Luo, Y.: Graph convolutional networks for text classification. In Proceedings of the AAAI, volume 33, pages 7370–7377, 2019.

Yao, S., Wang, T., and Wan, X.: Heterogeneous graph transformer for graph-to-sequence learning. In Proceedings of the Conference of Association for Computational Linguistics, pages 7145–7154, 2020.

Ye, Y. and Ji, S.: Sparse graph attention networks. arXiv preprint arXiv:1912.00552, 2019.

Yuan, Z. and Briscoe, T.: Grammatical error correction using neural machine translation. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 380–386, 2016.

Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J.: Graph transformer networks. In Proceedings of the Conference of Neural Information Processing Systems, pages 11983–11993, 2019.

Zhang, H., Liu, Z., Xiong, C., and Liu, Z.: Grounded conversation generation as guided traverses in commonsense knowledge graphs. In Proceedings of ACL, pages 2031–2043, 2020.

Zhang, L., Ge, Y., and Lu, H.: Hop-hop relation-aware graph neural networks. arXiv preprint arXiv:2012.11147, 2020.

Zhang, X. and Lapata, M.: Sentence simplification with deep reinforcement learning. In EMNLP, pages 584–594, 2017.

Zhang, X., Wei, F., and Zhou, M.: Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. In Proceedings of the Conference of Association for Computational Linguistics, page 5059–5069, 2019.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q.: Ernie: Enhanced language representation with informative entities. In Proceedings of ACL, pages 1441–1451, 2019.

Zhang, Z., Wu, Y., Zhao, H., Li, Z., Zhang, S., Zhou, X., and Zhou, X.: Semantics-aware bert for language understanding. In Proceedings of AAAI, 2020.

Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., and Huang, X.: Extractive summarization as text matching. In Proceedings of the Conference of Association for Computational Linguistics, 2020.

Zhou, J. and Keung, P.: Improving non-autoregressive neural machine translation with monolingual data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1893–1898, Online, July 2020. Association for Computational Linguistics.

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., and Zhao, T.: Neural document summarization by jointly learning to score and select sentences. In Proceedings of the Conference of Association for Computational Linguistics, page 654–663, 2018.

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., and Yu, Y.: Texygen: A benchmarking platform for text generation models. In Proceedings of SIGIR, pages 1097–1100, 2018.

# VITA

**Name**: Ye Liu

**EDUCATION**:

- B.E., in Computer Science and Technology, Northeastern University, 2015
- M.S., in Electrical and Computer Engineering, University of Illinois at Chicago, 2016

**PUBLICATIONS**:

- **Ye Liu**, Kazuma Hashimoto, Yingbo Zhou, Caiming Xiong and Philip S. Yu "Dense Hierarhical Retriever on Open-domain Question Answering." In Proceedings of EMNLP findings, 2021
- **Ye Liu**, Jianguo Zhang, Yao Wan, Congying Xia, Lifang He and Philip S. Yu. "HetFormer: Heterogeneous Transformer with Sparse Attention For Long-text Extractive Summarization." In Proceedings of EMNLP, 2021.
- Wenting Zhao, **Ye Liu**, Yao Wan and Philip S. Yu. "Attend, Memorize and Generate: Towards Faithful Table-to-Text Generation in Few Shots." In Proceedings of EMNLP findings, 2021
- Jingfeng Zhang, Haiwen Hong, Zhi Li, Yin Zhang, Yao Wan, **Ye Liu** and Yulei Sui. "Multi-Lingual Code Semantics Disentangle via Variational Auto-Encoder with Cross-Training." In Proceedings of ACL Findings, 2021.

- **Ye Liu**, Yao Wan, Lifang He, Peng Hao and Philip S Yu. "KG-BART: Knowledge Graph-Augmented BART for Generative Commonsense Reasoning." In Proceedings of AAAI, 2021.
- **Ye Liu**, Yao Wan, Jianguo Zhang, Wenting Zhao and Philip S Yu. "Enriching Non-Autoregressive Transformer with Syntactic and Semantic Structures for Neural Machine Translation." In Proceedings of EACL, 2021.
- Jianguo Zhang, Kazuma Hashimoto, Yao Wan, **Ye Liu**, Caiming Xiong and Philip S. Yu. "Are Pretrained Transformers Robust in Intent Classification?A Missing Ingredient in Evaluation of Out-of-Scope Intent Detection." https://arxiv.org/abs/2106.04564.
- **Ye Liu**, Tao Yang, Zeyu You, Wei Fan and Philip S. Yu. "Commonsense Evidence Generation and Injection in Reading Comprehension." In Proceedings of SIGDIAL, 2020.
- **Ye Liu**, Chenwei Zhang, Xiaohui Yan and Philip S. Yu. "Generative Question Refinement with Deep Reinforcement Learning in Retrieval-based QA System." In Proceedings of CIKM, 2019.
- Jianguo Zhang, Pengcheng Zou, Zhao Li, Yao Wan, **Ye Liu**, Xiuming Pan, Yu Gong and Philip S. Yu. "Product Title Refinement via Multi-Modal Generative Adversarial Learning." In Proceedings of NIPS Workshop, 2018.
- **Ye Liu**, Shaika Chowdhury, Chenwei Zhang, Cornelia Caragea and Philip S Yu. "Interpretable Multi-Step Reasoning with Knowledge Extraction on Complex Healthcare Question Answering." https://arxiv.org/abs/2008.02434

- **Ye Liu**, Lifang He, Bokai Cao, Philip S. Yu, Ann B Ragin and Alex Leow. "Multi-View Multi-Graph Embedding for Brain Network Clustering Analysis." In Proceedings of AAAI, 2018.
- **Ye Liu**, Jiawei Zhang, Chenwei Zhang and Philip S. Yu. "Data-driven Blockbuster Planning on Online Movie Knowledge Library." In Proceedings of IEEE BigData, 2018.