Neural Text Generation from Structured and Unstructured Data

by

Hamidreza Shahidi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

A number of researchers have recently questioned the necessity of increasingly complex neural network (NN) architectures. In particular, several recent papers have shown that simpler, properly tuned models are at least competitive across several natural language processing tasks. In this thesis, we show that this is also the case for text generation from structured and unstructured data. Specifically, we consider neural table-to-text generation and neural question generation (NQG) tasks for text generation from structured and unstructured data respectively. Table-to-text generation aims to generate a description based on a given table, and NQG is the task of generating a question from a given passage where the generated question can be answered by a certain sub-span of the passage using NN models. Experiments demonstrate that a basic attention-based sequence-to-sequence model trained with exponential moving average technique achieves state of the art in both tasks. We further investigate using reinforcement learning with different reward functions to refine our pre-trained model for both tasks.

Acknowledgements

I would like to thank my advisors, Professor Ming Li and Professor Jimmy Lin for their invaluable support and guidance. I am also grateful to Professor Pascal Poupart and Professor Yaoliang Yu for accepting to review this thesis as the members of my committee. My special thanks are extended to my friends at RSVP.ai company for making my research an enjoyable experience and providing me with their resources. Finally, I wish to thank my family, especially my parents who have sacrificed a lot to raise me up and to tolerate the hardship of me being away, for their unconditional support throughout my life.

Dedication

This is dedicated to my parents for their endless love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent natural language processing (NLP) literature can be characterized as increasingly complex neural network architectures that eke out progressively smaller gains over previous models. Following a previous line of research [1, 46, 48], we investigate the necessity of such complicated neural architectures. In this thesis, our focus is on text generation from structured and unstructured data by considering description generation from a given table (table-to-text) and question generation from a given passage and target answer (text-to-text).

More specifically, the goal in the neural table-to-text generation task is to generate biographies based on Wikipedia infoboxes (structured data). An infobox is a factual table with a number of fields (e.g., name, nationality, and occupation) describing a person. For this task, we use the WIKIBIO dataset [37] as the benchmark dataset. Figure 1.1 shows an example of a biographic infobox form this dataset.

Automatic question generation aims to generate a syntactically correct, semantically meaningful and relevant question from a natural language text and a target answer within it (unstructured data). This is a crucial yet challenging task in NLP which has received growing attention due to its applications in improving question answering systems [15, 65, 66], providing material for educational purposes [23], and helping conversational systems to start and continue a conversation [49]. We adopt the widely used SQuAD dataset [55] for this task. Table 1.1 illustrates one instance of (passage, answer, question) triplets in this dataset.

Prior works have made remarkable progress on both of these tasks. However, the proposed models utilize complex neural architectures to capture the necessary information of the input(s). In this work, we question the need for such sophisticated NN models for text generation from inputs with structured and unstructured data.

	Bernard Keen
Born	September 5, 1890
Died	August 5, 1981 (aged 90)
Nationality	British
Awards	Fellow of the Royal Society[1]
	Scientific career
Fields	Soil scientist
Institutions	University College London

Figure 1.1: An example infobox from the WIKIBIO dataset. The corresponding description for this infobox is "Sir Bernard Augustus Keen FRS (5 September 1890 – 5 August 1981) was a British soil scientist and Fellow of University College London.".

Passage: Hydrogen is commonly used in power stations as a coolant in generators due to a number of favorable properties that are a direct result of its light diatomic molecules.
Answer: as a coolant in generators
Question: How is hydrogen used at power stations?

Table 1.1: A sample (passage, answer, question) triplet from the SQuAD dataset.

Particularly, we adopt a bi-directional, attention-based sequence-to-sequence (seq2seq) model [62] equipped with copy mechanism [19] *for both tasks*. We demonstrate that this model together with exponential moving average (EMA) technique achieves state of the art in both neural table-to-text generation and NQG. Interestingly, our model is able to achieve this result without even using any linguistic features.

1.1 Task Definition

Let us formally denote the input sequence of length N as $X = (x_1, x_2, \ldots, x_N)$ and the output sequence of length M as $Y = (y_1, y_2, \ldots, y_M)$. Generally, text generation from input X is defined to find Y^* such that:

$$Y^* = \underset{Y}{argmax} \prod_{t=1}^{M} P(y_t | y_{1:t-1}, X; \theta) \tag{1.1}$$

Here, θ is the model parameters, X is the field-value records (i.e. content and field representation) for neural table-to-text generation, and the passage and the target answer for NQG. Note that a passage can be the whole paragraph or the sentence(s) containing the target answer. In this work, we use the latter as the input of our encoder.

In summary, the goal in table-to-text generation is to generate the most likely biography description given the corresponding infobox. Additionally, NQG aims at generating the most likely question for a given passage and an answer within the passage.

1.2 Contributions

Our contributions are three-fold: (1) We propose a *unified* NN model for text generation from structured and unstructured data built on an attention-based seq2seq model equipped with copy mechanism. We show that training this model with the EMA technique leads to the state of the art in neural table-to-text generation as well as NQG. (2) Because our model is in essence the main building block of previous models, our results show that some previous works espouse needless complexity, and that gains from previously-proposed complex neural architectures are quite modest. In other words, state of the art is achieved by careful tuning of simple and well-engineered models, not necessarily by adding more complexity to the model [39]. To provide a solid foundation for future efforts, all code for this work will be released under an open-source license. (3) We investigate the performance of our model after using reinforcement learning (RL) with different reward functions (e.g., BLEU, GLEU, and ROUGE score) as a way to refine our pre-trained model for both tasks.

1.3 Thesis Organization

In Chapter 2, we discuss related work for neural table-to-text generation and NQG and use their results to compare against ours in the rest of the thesis. Chapter 3 introduces a simple yet effective model for both tasks. We further discuss training this model with maximum likelihood estimation (MLE) and refining it with reinforcement learning. Chapter 4 presents how we conduct our experiments and shows experimental results on the WIKIBIO and SQuAD datasets. In Chapter 5, we provide a conclusion and summary for the thesis and discuss some potential research directions for the future.

Chapter 2

Background and Related Work

NLP tasks can be divided into four main categories based on the input and output types: sentence classification (e.g., sentiment analysis), sentence pair classification (e.g., sentence similarity and paraphrase identification), sequence labeling (e.g., named entity recognition and part of speech tagging), and sequence-to-sequence generation (e.g., translation, text summarization, and dialogue generation). In this thesis, our focus is on sequence-to-sequence generation. Particularly, we consider table-to-text generation and text-to-text generation as two sub-problems of this category and investigate using a simple yet effective sequence-to-sequence model *for both tasks*. In what follows, we would first discuss sequence-to-sequence learning and reinforcement learning, then we specifically talk about the previously-proposed models in the literature for neural table-to-text generation and NQG.

2.1 Sequence-to-sequence Learning

Sequence-to-sequence learning [4, 10, 18, 67, 71] is a challenging task in artificial intelligence which has attracted much attention in recent years due to its application in neural machine translation (NMT) [3, 4, 10, 18, 35, 36, 67, 71], text summarization [18, 64], question answering [73], and conversational response generation [59, 68].

A major approach for sequence-to-sequence learning is the encoder-decoder framework: As depicted in figure 2.1, the encoder takes the source sequence $X = (x_1, x_2, \ldots, x_N)$ as input and generates a set of representations; the decoder uses the source representations and the preceding tokens to generate the target sequence $Y = (y_1, y_2, \ldots, y_M)$ one at a time by estimating the conditional probability of the current target token $P(y_t|y_{1:t-1}, X; \theta)$. Here, θ is the model parameters

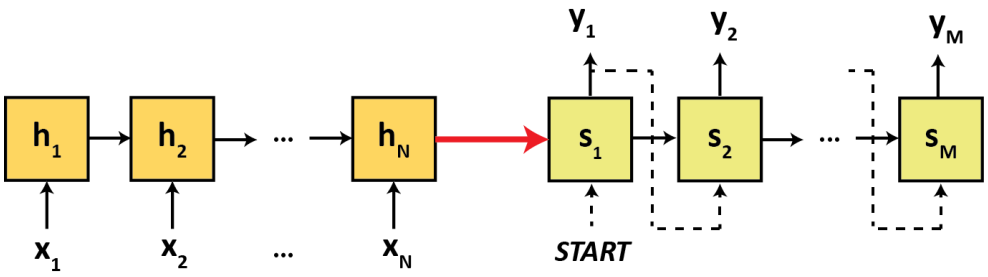

Figure 2.1: A sequence to sequence model.

and is usually learned by using log likelihood as the objective function:

$$L(\theta) = \sum_{(X,Y)\in(\mathcal{X},\mathcal{Y})} \log P(Y|X;\theta) \tag{2.1}$$

where $(\mathcal{X}, \mathcal{Y})$ are source and target pairs from the dataset. The conditional probability $P(Y|X;\theta)$ can be further factorized according to the chain rule:

$$P(Y|X;\theta) = \prod_{t=1}^{M} P(y_t|y_{1:t-1}, X;\theta) \tag{2.2}$$

where $y_{1:t-1}$ is the preceding tokens before position t. Moreover, Bahdanau et al. [4] introduce attention mechanism between the encoder and decoder to find which source representation to focus on when predicting the current token according to its relevance to different parts of the source sequence. Note that the attention weights are learned during training.

LSTM units [25] are typically used in the seq2seq models. The following equations summarize mathematical operations inside an LSTM unit:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \end{bmatrix} = \sigma\left(\begin{bmatrix} W_i \\ W_f \\ W_o \end{bmatrix} [h_{t-1}, x_t]\right) \tag{2.3}$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t]) \tag{2.4}$$

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t \tag{2.5}$$

$$h_t = o_t \times \tanh(c_t) \tag{2.6}$$

where i_t, f_t, and o_t are input gate, forget gate, and output gate respectively. Hidden state and cell state at time step t are indicated by h_t and c_t, and σ is the sigmoid non-linearity. Note that W_i, W_f, W_o, W_c are all weight matrices.

2.2 Reinforcement Learning

In this section, we briefly explain the relevant concepts and algorithms in reinforcement learning that we are going to use:

Markov Decision Process

Initially, we define a Markov decision process (MDP). An MDP provides a mathematical framework for modeling sequential decision making and formally describes a fully observable environment for reinforcement learning. MDPs are essentially a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, where $\mathcal{S}$ and $\mathcal{A}$ are a finite set of states and actions respectively. In our setting, actions (choosing words from the vocabulary) are discreet, but they can be continuous in general. In addition, $\mathcal{P} : \mathcal{A} \times \mathcal{S} \times \mathcal{S} \to [0, 1]$ is a state transition probability function, $\mathcal{R} : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1)$ is a discount factor.

Reinforcement Learning

The goal of reinforcement learning is to find the best policy such that the agent would maximize its sum of discounted rewards, $G = \Sigma_{t=0}^{\infty} \gamma^t r_{t+1}$, by following this policy. A policy π is a probability distribution over actions conditioned on states, $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$, and fully defines the behaviour of an agent. The state-value function V and the action-value function Q of a policy π are defined as:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] \tag{2.7}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \tag{2.8}$$

where $G_t = \Sigma_{k=0}^{\infty} \gamma^k r_{t+k+1}$. They can also be expressed by the following expressions (a.k.a Bellman Expectation Equation):

$$V_\pi(s) = \mathbb{E}_\pi[r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s] = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V_\pi(s')] \tag{2.9}$$

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a] \\ &= \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma \sum_{a' \in \mathcal{A}} \pi(a'|s') Q_\pi(s', a')] \end{aligned} \tag{2.10}$$

where $\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$ and $\mathcal{R}^a_{ss'} = \mathbb{E}[r_{t+1} | s_t = s, s_{t+1} = s', a_t = a]$. That being said, in reinforcement learning, we wish to find the optimal policy π^* in an unknown environment (i.e., with an unknown MDP). This policy corresponds to $V_* = \max_\pi V_\pi$ and $Q_* = \max_\pi Q_\pi$ value functions.

Policy Gradient

In policy gradient methods [63], the policy $\pi(a|s;\theta)$ is parameterized by θ which is updated by optimizing the objective function $J(\theta) = \mathbb{E}_{\pi_\theta}[\Sigma_{t=0}^{\infty} \gamma^t r_{t+1}]$ using gradient ascent.

Monte Carlo policy gradient

Williams [70] proposes the Monte Carlo policy gradient algorithm (a.k.a REINFORCE) to compute the derivative of the objective function $J(\theta)$ by using the likelihood ratio method according to the following expression:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log\ \pi_\theta(a_t|s_t)\ G_t] \tag{2.11}$$

We can also reduce the variance by subtracting a baseline function $b(s_t)$ from G_t,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log\ \pi_\theta(a_t|s_t)\ (G_t - b(s_t))]. \tag{2.12}$$

We further approximate the above expectation by an unbiased sample and update the weights using the following formula:

$$\Delta\theta_t \approx \alpha \nabla_\theta \log\ \pi_\theta(a_t|s_t)\ (G_t - b(s_t)) \tag{2.13}$$

where α is the learning rate.

Actor-Critic Policy Gradient

Monte Carlo policy gradient still has high variance. To reduce the variance, Konda and Tsitsiklis [32] propose a critic $Q_w(s,a) \approx Q_{\pi_\theta}(s,a)$ instead of G_t, and use $V_v(s) \approx V_{\pi_\theta}(s)$ as the baseline. Here, w and v are the parameters of the action-value and state-value function approximators respectively, and they are updated during the training procedure. We define the advantage function as $A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t)$ and update the weights of the policy according to the following expression:

$$\Delta\theta_t \approx \alpha \nabla_\theta \log\ \pi_\theta(a_t|s_t)\ A(s_t, a_t) \tag{2.14}$$

In practice, we use temporal difference error as an unbiased estimate of the advantage function to reduce the number of required parameters. Therefore, the updates are actually performed based on what follows:

$$\Delta\theta_t \approx \alpha\nabla_\theta \log\ \pi_\theta(a_t|s_t)\ (r_t + \gamma V_v(s_{t+1}) - V_v(s_t)) \tag{2.15}$$

2.3 Neural Table-to-text Generation Models

Traditionally, most table-to-text generation systems have been separated into individual modules: (1) a module was responsible to select a subset of relevant information in the input data (content selection), (2) another module was used to generate natural language descriptions based on the output of the first module (surface realization). To learn the first module, Barzilay and Lapata [6], and Duboue and McKeown [16] propose a content selection model which aligns records and sentences. Liang et al. [38] propose a hierarchical hidden semi-Markov model which first chooses relevant information and then generates a description based on the chosen facts. Later works [2, 28, 33] suggest combining these modules together.

More recently, Mei et al. [45] use an encoder-decoder style NN model for selective generation on the WEATHERGOV and ROBOCUP datasets. While these datasets contain only a few tens of thousands of records, Lebret et al. [37] introduce WIKIBIO dataset that contains over 700k biographies from Wikipedia. They also propose an n-gram statistical language model with local and global conditioning which considers both the content and the structure of the table. Since an n-gram language model makes an order n Markov assumption, Equation 2.2 changes to what follows:

$$P(Y|X;\theta) = \prod_{t=1}^{M} P(y_t|y_{t-(n-1):t-1}, X;\theta) \tag{2.16}$$

where X contains both local information (i.e., information about the occurrence of the previously-generated words in the table) and global information (i.e., information about all tokens and fields of the table regardless whether they have appeared in the previously-generated words or not). Their *neural language model* (*NLM*) considers both the content and the structure of the table by using field and position embeddings. However, *NLM* only uses local conditioning without copying actions (i.e. with fixed vocabulary). They also introduce the following baselines for comparison:

- *The Kneser-Ney* (*KN*) model is a language model proposed by Heafield et al. [21].

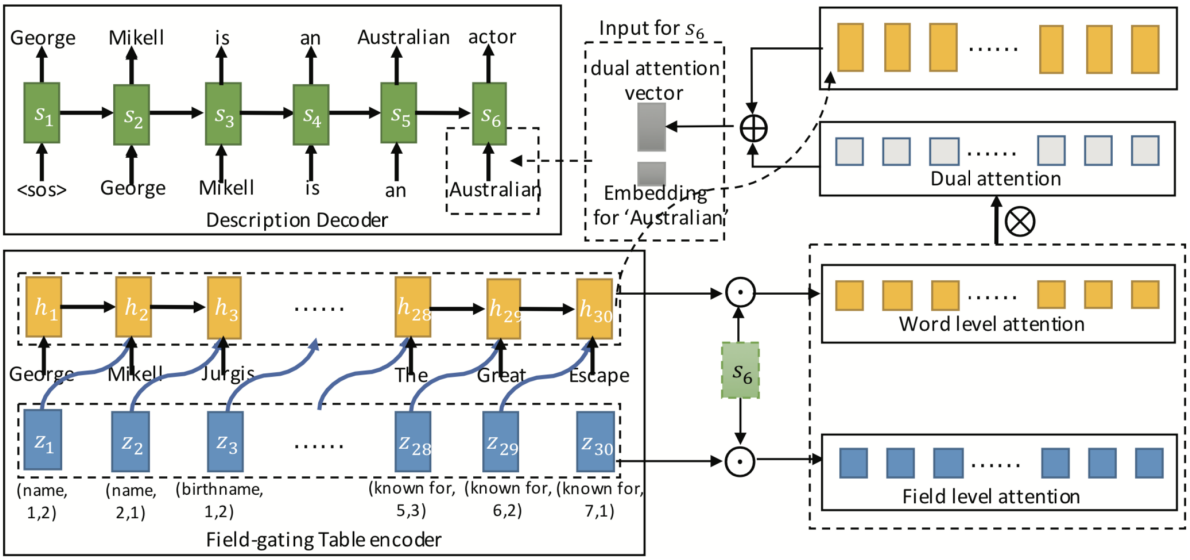

Figure 2.2: The overall architecture of the structure-aware seq2seq model. The figure is copied from Liu et al. [43].

- *Template KN* learns a KN language model over templates by replacing the words occurring in both the table and the training sentences with a special token.

The most competitive statistical language model proposed by Lebret et al. [37] (*Table NLM*) incorporates both local and global conditioning and also employs copy mechanism. However, even their ultimate model is not effective enough to capture long-range contextual dependencies while generating descriptions.

To deal with this issue, Liu et al. [43] suggest a structure-aware seq2seq (*Struct-aware*) architecture with specific methods to effectively conduct local and global addressing on the table. While local addressing is realized by content encoding of the model's encoder and word level attention, global addressing is realized by field encoding of the field-gating LSTM variation and field level attention.

More specifically, in the encoding phase, field gating mechanism is introduced to consider field information while updating the cell memory of the LSTM units. That being said, Equation 2.5 changes according to the following expressions:

$$l_t = \sigma(W_l \times z_t) \tag{2.17}$$

$$\hat{z}_t = tanh(W_z \times z_t) \tag{2.18}$$

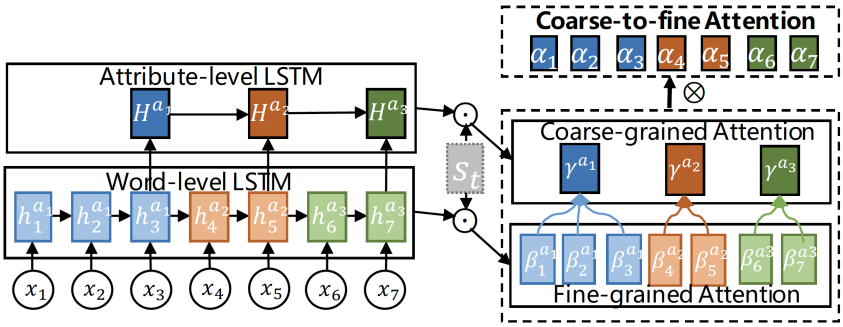

Figure 2.3: The overall architecture of the seq2seq model with hierarchical encoder and auxiliary supervision. The figure is adopted from Liu et al. [42].

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t + l_t \times \hat{z}_t \tag{2.19}$$

where z_t is the field embedding of the current token, and W_l and W_z are trainable model parameters. Furthermore, in the decoding phase, they use dual attention mechanism which incorporates both word level and field level attention to better capture the semantic relevance between the generated description and the table information. Figure 2.2 illustrates the architecture of this model more concisely.

Bao et al. [5] propose *Table2Seq* which is a seq2seq model equipped with a specifically designed copy mechanism that selectively replicates contents from the table to the output sequence. Their model takes a row from the table and generates a natural language sentence describing that row by leveraging the semantics of the table. Furthermore, Sha et al. [58] argue that when a human writes a description based on a given table, he or she would probably consider the content order before wording. They propose an order-planning text generation model that is able to better capture the structural information of the table to make the generated text more fluent and smooth.

To model the field-value structure of a tables, Liu et al. [42] propose a two-level hierarchical encoder with coarse-to-fine attention as shown in Figure 2.3. They also propose three joint tasks (auxiliary sequence labeling task, text autoencoder, and multi-labeling classification) as auxiliary supervision to capture the accurate semantic representation of the tables. The auxiliary sequence labeling task predicts the field names based on the hidden states of the field-level LSTM to enforce the encoder to remember the structure of the table. The text autoencoder uses the internal representation of an auto-encoder trained to reconstruct the descriptions to supervise that of the hierarchical encoder by minimizing their distance. Multi-labeling classification aims to predict all the field names which appear in the table as the target using the input representation constructed by the hierarchical encoder.

In this thesis, we study the necessity of such sophisticated NN models. Particularly, similar to Lebret et al. [37], we use both content and field information to represent a table by concatenating field and position embeddings to the word embedding at each time step. Unlike Liu et al. [43], we don't separate local and global addressing by adopting specific modules for each, but rather adopt the EMA technique and let the bi-directional model to do it implicitly using natural advantages of the model.

2.4 Neural Question Generation Models

Previous NQG models can be classified into two main categories: rule-based approaches and neural-network-based approaches. Rule-based methods [47, 57] make use of manually designed rules and templates which are extremely costly to collect and can not generalize well to unseen instances. For instance, *PCFG-Trans* proposed by Heilman [22] is a rule-based system that can generate question based on a given word span.

On the other hand, Du et al. [14] propose using a seq2seq model for question generation called *s2s-ans* and achieves better results than rule-based systems. Their model only encodes the passage and doesn't take the target answer into account. In contrast, Zhou et al. [76] concatenate answer position indicator with the word embedding to make the model aware of the target answer. Additionally, they use lexical features (e.g., POS and NER tags) to enrich their model's encoder. They also deploy a copy mechanism to enable their model to copy input words to the output. Their ultimate model is indicated by *NQG++* in the following sections.

However, Song et al. [60] argue that adopting a single bit to indicate answer position is not sufficient to capture the context information. They suggest using multi-perspective context matching algorithm [69] on top of BiLSTM outputs to leverage the information from explicit interaction between the passage and the target answer. Their goal is to figure out whether each passage word belongs to the relevant context of the target answer. Towards this, they adopt three different strategies to match the passage with the target answer. Each strategy constructs a matching vector by using a different representation for the target answer listed below:

- *Full-matching* considers the last hidden state of the answer encoder which takes words order into account.
- *Attentive-matching* uses a weighted sum of all answer states without considering words order as the answer representation.

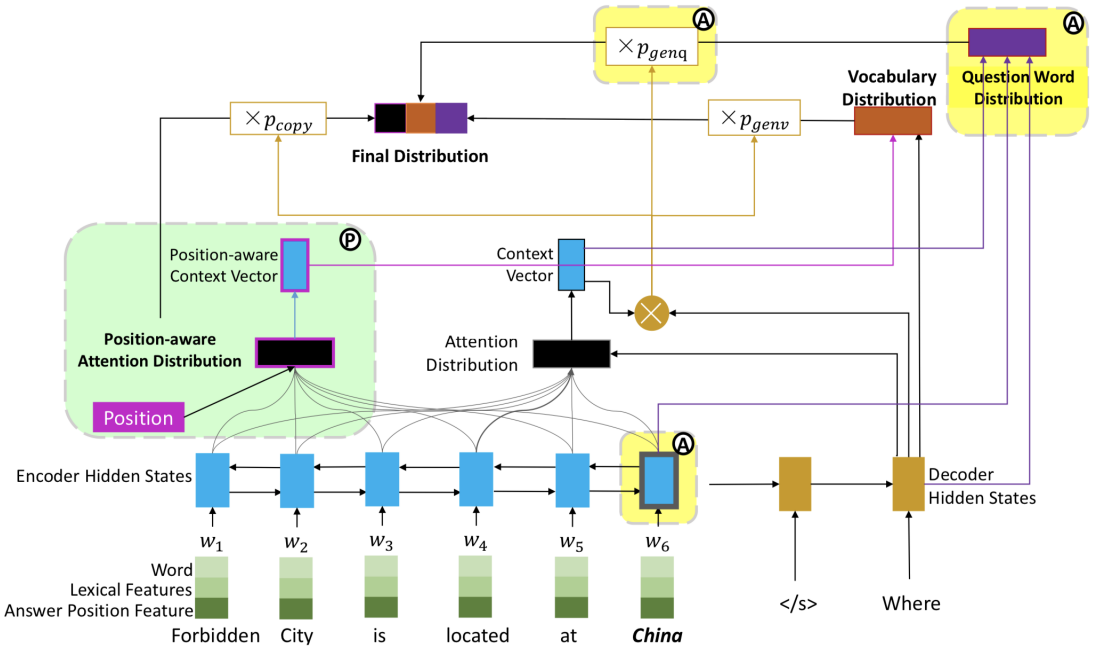

Figure 2.4: The overall architecture of the answer-focused and position-aware NQG model. The figure is adopted from Sun et al. [61].

- *Max-attentive-matching* constructs the answer representation by taking the maximum element on each dimension of answer states. This strategy considers the most relevant answer state to the passage state.

For all strategies, each dimension of the matching vector is computed by multiplying both input vectors (i.e., answer representation and passage state) by some tunable weights and calculating their cosine similarity. The final matching vector is the concatenation of the matching results of all three strategies. They also employ another BiLSTM layer over the matching vectors and concatenate its hidden states with the hidden states of the passage encoder to be used in the attention mechanism. Their proposed model is called *MPQG*.

In addition, Sun et al. [61] propose an answer-focused and position-aware model (*AnsPos-aware*). Their model incorporates the answer embedding to avoid generating question words that do not match the answer type (answer-focused). They also deploy relative distance between the context words and the target answer to ensure that the model does not copy the context words that are far from and irrelevant to the target answer (position-aware). Their model is depicted in Figure 2.4.

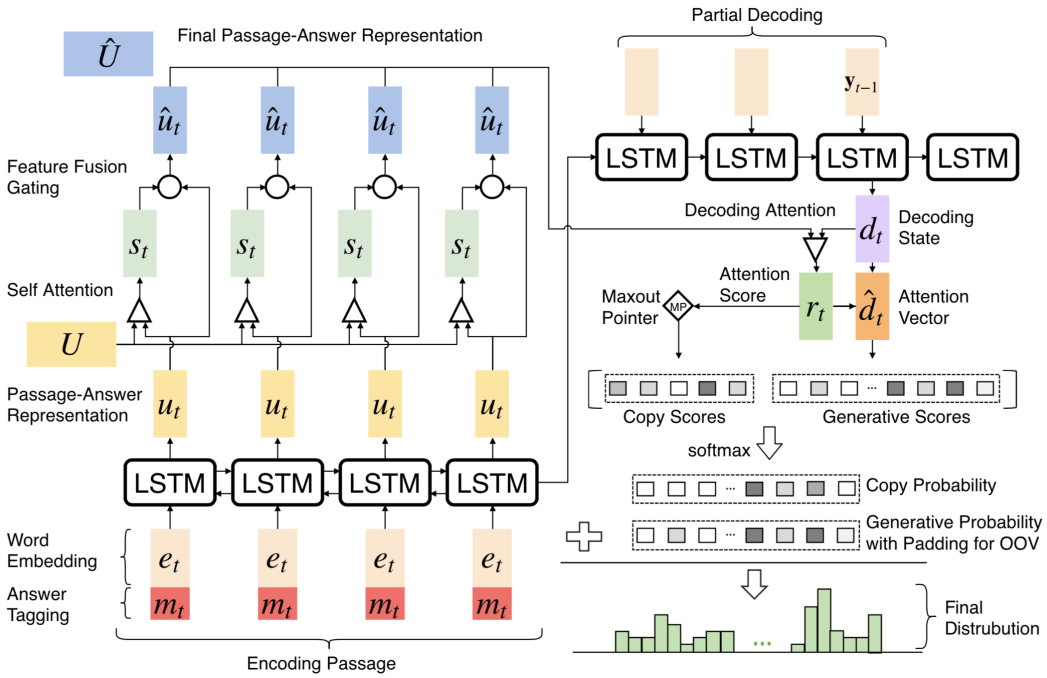

Figure 2.5: The overall architecture of the the paragraph-level NQG model with maxout pointer and gated self-attention. The figure is adopted from Zhao et al. [75].

Most NQG models only use one or two sentences of the input paragraph that include the target answer as the input passage. Zhao et al. [75] state that using the whole paragraph as context enables the model to generate high quality questions. However, performance drops when using the whole paragraph as the input in a naive way. This is mainly because of the various challenges that long text poses for seq2seq models in question generation. Hence, the authors propose a maxout pointer mechanism with gated self-attention encoder (*s2s-a-at-mp-gsa*) to address the challenges of processing long text inputs for question generation. Maxout pointer is an extension of existing copy mechanisms that prevents repetitions in the output sequence by limiting the probability of copying repeated words to their maximum value. Moreover, gated self-attention encoder is designed to effectively utilize relevant information at paragraph-level. Figure 2.5 provides an overview of their model in more detail.

More recently, Kim et al. [29] observe that a significant portion of the generated questions include words from the target answer and use answer-separated seq2seq (*ASs2s*) which replaces the target answer in the passage with a special token to avoid using the answer words in the generated question. They also utilize keyword-net to extract key information from the target

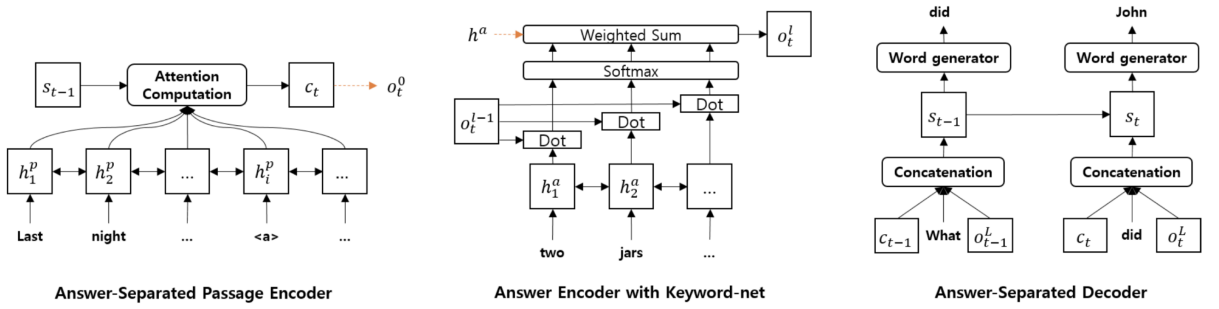

Figure 2.6: The overall architecture of the answer-separated seq2seq model. The figure is adopted from Kim et al. [29].

answer. Figure 2.6 illustrates each component of their model.

Similarly, Liu et al. [41] propose Guided Copy Network for Question Generation (CGC-QG). They use a clue word predictor by adopting graph convolution networks which have been used in different NLP tasks [12, 31, 40, 44, 74] to highlight the imperative aspects of the input passage. Namely, they deploy graph convolutional networks on dependency tree of each passage to find clue words in the input. The prediction made by the clue word predictor is given as a feature to the passage encoder to help identify whether each word in the input passage is a potential clue to be copied into the question or not. Figure 2.7 illustrates the architecture of their model.

A number of previous works [26, 34] use reinforcement learning to directly optimize task-specific scores including evaluation metrics such as BLEU and ROUGE for NQG. For instance, Kumar et al. [34] adopt deep reinforcement learning framework to refine a pre-trained model. Their best model is denoted by RL_{ROUGE} which uses ROUGE score as the reward function.

Finally, it is worth mentioning other interesting research directions for this task in the literature. Tang et al. [65], Tang et al. [66], and Duan et al. [15] model question answering and question generation as dual tasks and use question generation to improve the performance of question answering system. Gao et al. [17] take difficulty into consideration and generate questions under the control of specified difficulty levels. Yao et al. [72] propose *s2s+z+c+GAN* which employs GAN framework to capture the variability in questions using a latent variable and generate certain types of questions by introducing an additional observed variable. Furthermore, Zhou et al. [77] suggest *SeqCopyNet* for question generation, a method for seq2seq models to copy a sequence of words rather than a single word.

Our model is architecturally more similar to Zhou et al. [76] with the following distinctions: (1) we do not use lexical features (i.e. POS tag, NER tag, and word case feature), (2) we utilize

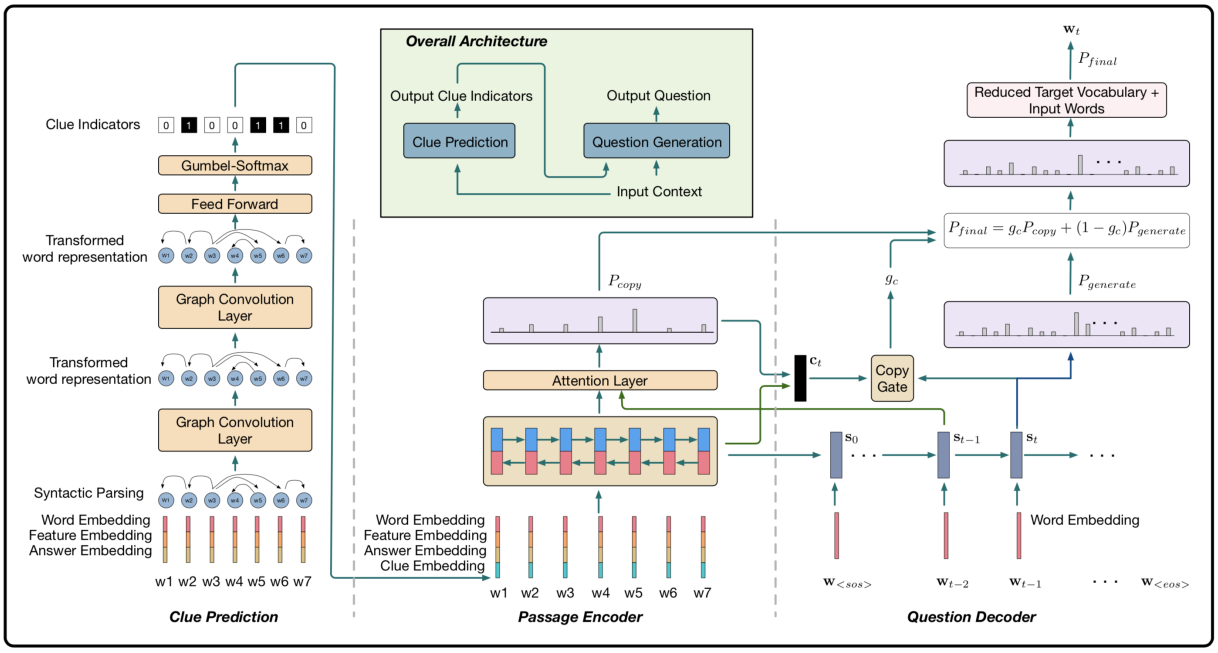

Figure 2.7: The overall architecture of the *CGC-QG*. The figure is adopted from Liu et al. [41].

EMA technique while training and use the averaged weights for evaluation, (3) we do not make use of the introduced maxout hidden layer, (4) we adopt LSTM units instead of GRU units. As shown in section 4.4, these distinctions along with some hyperparameter differences, notably the optimizer and learning rate, have had a huge impact on the experimental results.

Chapter 3

Proposed Model

In this chapter, we introduce a simple, unified, attention-based seq2seq model for both neural table-to-text generation and NQG. Figure 3.1 illustrates an overview of our model.

3.1 Encoder

The encoder is a bi-directional LSTM (BiLSTM) whose input x_t at time step t is the concatenation of the current word embedding e_t with some additional task-specific features. For instance, in Figure 1.1, we flatten the table and treat it as a plain sequence ("Bernard", "Keen", "September", "5", "1890", "August", "5", "1981", ...). Thus, e_1 is the embedding of "Bernard", e_2 is the embedding of "Keen", and so on. Similarly, in Table 1.1, the sequence is ("Hydrogen", "is", "commonly", "used", "in", "power", "stations", ...). Therefore, e_1 is the embedding of "Hydrogen", e_2 is the embedding of "is", and so on.

For table-to-text generation, additional features are field name f_t and position information p_t following Lebert et al. [37]. The position information itself is the concatenation of p_t^+ which is the position of the current word in its field when counting from left and p_t^- when counting from right. Considering the word *University* in Figure 1.1 as an example, it is the first word from left and the third word from right in the *Institutions* field. Hence, the structural information of this word would be {*Institutions*,1,3}. That being said, the input to the encoder at time step t for this task is:

$$x_t = [e_t; f_t; p_t^+; p_t^-] \tag{3.1}$$

We hypothesize that our model is able to utilize the information in p_t^+ and p_t^- more effectively as a result of deploying a bi-directional encoder. For NQG, similar to Zhou et al. [76], we use a

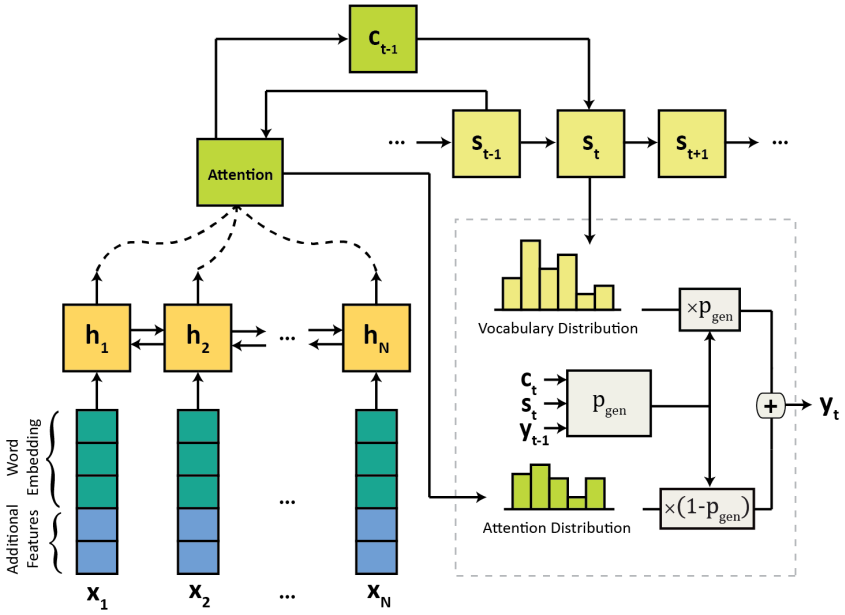

Figure 3.1: An overview of our model.

single bit b_t indicating whether the t[th] word in the passage belongs to the target answer or not as an additional feature. Hence, the input at time step t is:

$$x_t = [e_t; b_t] \tag{3.2}$$

Remarkably, unlike previous works [29, 60], we do not use a separate encoder for the target answer to have a *unified* model for both tasks. Each hidden state h_t of the encoder is the concatenation of a forward hidden state $\overrightarrow{h_t}$ and a backward hidden state $\overleftarrow{h_t}$:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(\overrightarrow{h_{t-1}}, x_t) \tag{3.3}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(\overleftarrow{h_{t+1}}, x_t) \tag{3.4}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{3.5}$$

Eventually, We initialize our decoder according to the following equation:

$$s_0 = tanh(W_{init}[\overleftarrow{h_0}; \overrightarrow{h_N}]) \tag{3.6}$$

where W_{init} is a weight matrix, and N is the number of input words.

3.2 Attention-based Decoder

Decoder is an attentional LSTM model [4]. The following equations summarize our decoder's operations:

$$s_t = \text{LSTM}([y_{t-1}; c_{t-1}], s_{t-1}) \tag{3.7}$$

$$e_{t,i} = v^\top \tanh(W_s s_t + W_h h_i + b_{att}) \tag{3.8}$$

$$a_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=0}^{N} \exp(e_{t,j})} \tag{3.9}$$

$$c_t = \sum_{i=0}^{N} a_{t,i} h_i \tag{3.10}$$

$$P^t_{vocab} = \text{softmax}(W_o[s_t; c_t] + b_o) \tag{3.11}$$

where s_t, y_t, c_t, and P^t_{vocab} are hidden state of the decoder, output word, context vector, and vocabulary distribution respectively all at time step t. v, W_s, W_h, b_{att}, W_o, and b_o are trainable parameters of the model.

Due to the considerable overlap between input and output words, we use a copy mechanism which integrates the attention distribution over the input words with the vocabulary distribution:

$$P^t_{final} = (1 - p_{gen})P^t_{att} + p_{gen}P^t_{vocab} \tag{3.12}$$

where $p_{gen} \in [0, 1]$ is a gate which determines whether to generate a word from the vocabulary or to copy it directly from the input text, and P^t_{att} is the current attention distribution. Particularly, p_{gen} can be expressed as

$$p_{gen} = \sigma(w_c^T c_t + w_s^T s_t + w_y^T y_{t-1} + b_{copy}), \tag{3.13}$$

where the vectors w_c, w_s, w_y, and the scalar b_{copy} are model parameters to be learned. While decoding, a special token $<sos>$ is used to begin the process. We terminate decoding whenever we get a special end of sentence token $<eos>$.

3.3 Exponential Moving Average

Training of NN models, especially RNNs, is a challenging task as it requires solving a high-dimensional non-convex optimization problem. This means that the final weights at the end of

training are not necessarily the best ones to use. One approach to address this problem is to use an average of the models observed towards the end of training by using a linearly or exponentially weighted average of the model weights, which is generally referred to as temporal averaging.

This technique has been initially introduced to be used in optimization algorithms for better generalization and reducing noise from stochastic approximation in recent parameter estimates. In particular, Moulines et al. [50] initially show that Polyak-Ruppert averaging [53, 56] improves the convergence of standard SGD. Alternatively, Kingma et al. [30] introduce an exponential moving average over the parameters, giving more weight to recent parameter values.

Similarly, we maintain two sets of parameters: (1) training parameters θ which are trained as usual, (2) evaluation parameters $\overline{\theta}$ that are exponentially weighted moving average of training parameters. The moving average is calculated using the following expression:

$$\overline{\theta} \leftarrow \beta \times \overline{\theta} + (1 - \beta) \times \theta \tag{3.14}$$

where β is the decay rate value. Evaluations that use the averaged parameters often produce more stable and accurate results than the final trained values. This is mainly because averaging the weights has the effect of stabilizing the optimization process that may be noisy due to the choice of hyperparameters (e.g., learning rate and optimizer) or the shape of the mapping function that is being learned. Obviously, the downside of this technique is that it requires keeping an extra copy of all trainable variables which could be expensive especially for large models. This simple training technique has had a huge effect in improving the performance of our model in both tasks (see Section 4.4).

3.4 Training Criteria

Given a dataset containing N input-output sequence pairs, denoted by $\mathcal{D} \equiv \{(X^{(i)}, Y^{*(i)})\}_{i=1}^{N}$, standard maximum likelihood training intends to maximize the sum of log probabilities of the ground-truth outputs given the corresponding inputs,

$$\mathcal{O}_{ML}(\theta) = \sum_{i=1}^{N} \log P(Y^{*(i)}|X^{(i)}, \theta). \tag{3.15}$$

When using the above maximum likelihood objective for training sequence generation models, a commonly used strategy is to use the ground-truth word at the previous time step as the input of the current time step rather than using the previous output of the model. This approach

which is called teacher forcing simplifies training the NN model to a one-step-ahead prediction task and improves convergence.

However, Bengio et al. [7] mention that teacher forcing causes a discrepancy between the training and inference stage, so-called *exposure bias*, in auto-regressive models. While, in training stage, the model predicts the current token according to the ground-truth information from the dataset, in inference stage, it has to predict the current token based on its own previous prediction. To alleviate this, the authors propose a training strategy called scheduled sampling, where the generative model is partially fed with its own predicted tokens rather than the ground-truth tokens during the training stage. Nonetheless, Huszar [27] show that this is an inconsistent training strategy and fails to address the problem fundamentally.

Another potential solution to address the above issue is to build the loss function on the entire generated sequence instead of each transition through reinforce learning. Furthermore, the ML objective (Equation 3.15) does not reflect task-specific scores such as BLEU and ROUGE. Several recent papers [34, 51, 71] have considered different ways of incorporating the task-specific score of a complete sequence generated without teacher forcing into the optimization of seq2seq models. In this work, we make use of the REINFORCE algorithm discussed in Section 2.2 and consider task-specific scores as the reward function. The following expression defines the expected reward objective:

$$\mathcal{O}_{RL}(\theta) = \sum_{i=1}^{N} \sum_{Y \in \mathcal{Y}} P(Y|X^{(i)}, \theta)\, r(Y, Y^{*(i)}) \tag{3.16}$$

where, $r(Y, Y^{*(i)})$ is the reward for the output sequence Y based on the ground-truth sequence $Y^{*(i)}$, and we are computing an expectation over all of the output sequences Y in the space of all candidate sequences $\mathcal{Y}$. As $\mathcal{Y}$ is a exponentially large space, it is impossible to compute the expectation exactly. Thus, we consider two strategies for sampling Y from $\mathcal{Y}$. The first strategy is greedy search in which we select the word with the highest probability at each time step. The second one is multinomial sampling [8]; in this strategy, we choose each word using multinomial sampling over the model's output distribution. The choice of different strategies reflects the exploration-exploitation dilemma in reinforcement learning. While greedy search strategy generates more accurate Y by exploiting the learned policy, multinomial sampling favours exploring more diverse candidates.

Note that it is a common practice in reinforcement learning to subtract a baseline (in our case the mean of rewards acquired till now) from $r(Y, Y^{*(i)})$ to reduce variance and increase stability (see Equation 2.12). Inspired from Wu et al. [71], to further stabilize training, we consider a linear combination of ML objective (Equation 3.15) and RL objective (Equation 3.16) according to what follows:

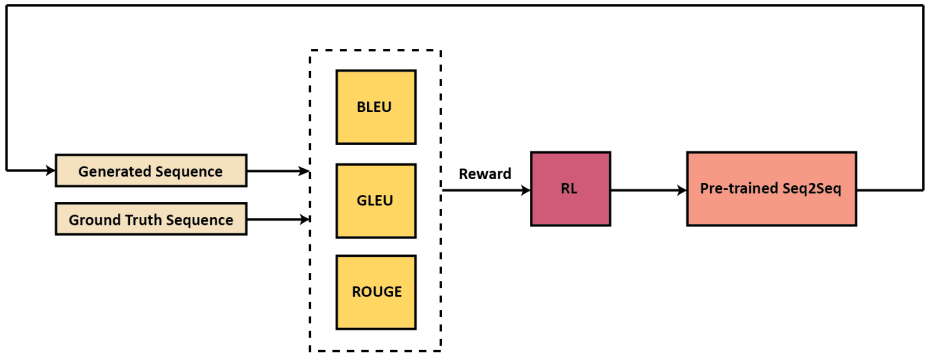

Figure 3.2: RL-refinement for both neural table-to-text generation and NQG with BLEU, GLEU, and ROUGE as the reward function.

$$\mathcal{O}_{Mixed}(\theta) = \alpha \times \mathcal{O}_{ML}(\theta) + \mathcal{O}_{RL}(\theta) \tag{3.17}$$

where α is a hyperparameter controlling the trade-off between ML and RL objectives.[1] In our setup, we first train a model using the ML objective until convergence. The results in Table 4.3 and 4.4 are obtained using the model that is only trained with the ML objective. Then, we use the mixed objective (Equation 3.17) to refine the pre-trained model. In Section 4.4, we compare the results from the pre-trained model with the results from the refined model.

3.5 Reward Functions

In this section, we discuss using reinforcement for refining our pre-trained model in both neural table-to-text generation and NQG, and experiment with several reward functions – BLEU, GLEU, and ROUGE – as a way to measure the quality of the generated sequence. Figure 3.2 illustrates the refinement procedure. As GLEU is not discussed in Section 4.2 as an evaluation metric, we explain it here.

GLEU is a score specifically designed to be used as reward in reinforcement learning framework by Wu et al. [71]. They mention that BLEU score has some undesirable properties when used for single sentences, as it was designed to be a corpus measure. Thus, they come up with a slightly different score for their RL experiments called GLEU. To calculate the GLEU score, they first record all 1, 2, 3 and 4-gram tokens in output and target sequences, and then they compute recall and precision. Recall is the ratio of the number of matching n-grams to the number

[1] We set α to 0.017 in our experiments.

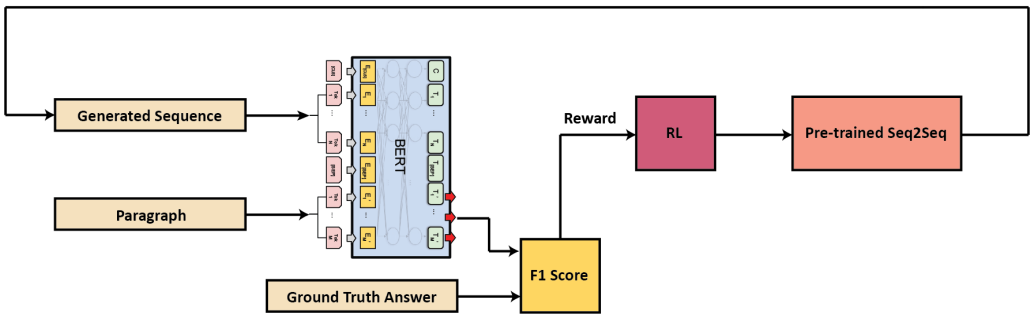

Figure 3.3: RL-refinement for NQG with BERT's F1 score as the reward function.

of total n-grams in the ground-truth sequence. Precision is the ratio of the number of matching n-grams to the number of total n-grams in the generated output sequence. Finally, GLEU score is defined as the minimum of recall and precision. The range of this score is always between 0 (no matches) and 1 (all match), and it is symmetrical when switching output and target sequences. Wu et al. [71] show that GLEU score correlates quite well with the BLEU score on a corpus level but does not have its drawbacks as a per sentence reward function.

Furthermore, for NQG, another approach to measure the quality of a generated question is to see whether it can be answered correctly by a question-answering (QA) system. We therefore feed the generated question together with its corresponding passage into a pre-trained QA system and use that system's F1 score as the reward. F1 score is a token-based measure proposed by Rajpurkar et al. [55] and accounts for partial word matches.

We use BERT (Bidirectional Encoder Representations from Transformers) [13] as our reference QA system. Broadly, it makes use of Transformer [67] and an attention mechanism that learns contextual relations between words. In contrast with directional models, which read the text input sequentially (left-to-right or right-to-left), BERT's encoder reads the entire sequence of words at once, and hence it is considered bi-directional. As a result, BERT model has been able to obtain state of the art results across a wide range of NLP tasks including question answering. Hence, we define the reward function according to the following expression:

$$r(Y, Y^*) = F1(A, A^*) \tag{3.18}$$

where $A = \text{BERT}(X, Y)$ and A^* is the ground-truth answer. Figure 3.3 depicts the setup for this experiment more concisely.

Chapter 4

Experimental Results

In this chapter, we first introduce the datasets, evaluation metrics, and implementation details. Then we present our experimental results and sample outputs for both neural table-to-text generation and NQG.

4.1 Datasets

We use the WIKIBIO dataset [37] for neural table-to-text generation. This dataset contains 728,321 articles from English Wikipedia, and uses the first sentence of each article as the ground-truth description of the corresponding infobox. The dataset has been divided into training (80%), test (10%), and validation (10%) sets. Table 4.1 summarizes the statistics of this dataset.

Avg. # tokens per sentence	26.1
Avg. # tokens per table	53.1
Avg. # table tokens per sentence	9.5
Avg. # fields per table	19.7

Table 4.1: WIKIBIO dataset statistics.

For NQG, the SQuAD dataset is used as the benchmark dataset. It contains 23,215 paragraphs from 536 articles with over 100k questions and their answers. The original dataset is divided into train and validation sets (the test split is held by the organizer for fair evaluation). That being said, Du et al. [14] and Zhou et al. [76] re-allocate it into train, validation, and test set which we will call split-1 and split-2 respectively. Data split-1 and split-2 contain 70,484/10,570/11,877

and 86,635/8,965/8,964 (train/validation/test) triplets respectively. In Table 4.4, we report experimental results on both data splits. Table 4.2 provides some statistics for this dataset.

Avg. # tokens per passage	32.7
Avg. # tokens per question	11.3
Avg. # tokens per answer	3.2

Table 4.2: SQuAD dataset statistics. Numbers are based on data split-2.

4.2 Evaluation Metrics

Following previous works, we compare the performance of neural table-to-text generation with BLEU-4 and ROUGE-4, and NQG with BLEU-4, METEOR, and ROUGE-L which are standard evaluation metrics for machine translation and text summarization.

- **BLEU-4:** BLEU-4 measures precision by counting the number of matching 4-grams in the candidate and the reference text.
- **METEOR:** METEOR is based on the harmonic mean of unigram precision and recall and considers exact, stem, synonym, and paraphrase matches.
- **ROUGE-4:** ROUGE-4 is a recall-oriented metric which measures the number of overlapping 4-grams between the candidate and the reference text.
- **ROUGE-L:** ROUGE-L compares the candidate and the reference text based on longest common sub-sequence shared between them.

To be consistent with previous works on each task, for neural table-to-text generation, we use NIST mteval-v13a.pl and rouge-1.5.5 to calculate BLEU and ROUGE respectively. With respect to NQG, we use the evaluation package released by Chen et al. [9] which was originally used to score image captions.

Furthermore, we report the mean and standard deviation (Mean $\pm$ Std) of each metric across multiple seeds to ensure robustness against potentially spurious conclusions [11].

4.3 Implementation Details

For the sake of reproducibility, we discuss the implementation details for achieving the results shown in Section 4.4. In Table 4.3 and 4.4, we train the model using cross-entropy loss, and keep the model which works best on the validation set during training for both tasks. We replace unknown tokens with a word from the input having the highest attention score $a_{t,i}$ according to Equation 3.9. In addition, a decay rate of 0.9999 is used for the exponential moving average in both tasks. In what follows, we first describe the hyperparameters used for the neural table-to-text generation task, and then the NQG task.

4.3.1 Hyperparameters for Neural Table-to-text Generation

For the neural table-to-text generation task, we train the model up to 10 epochs with batch size of 32. We use a single-layer BiLSTM for the encoder and a single-layer LSTM for the decoder and set the dimension of the LSTM hidden states to 500. Optimization is performed using Adam optimizer with a learning rate of 0.0005 and gradient clipping when its norm exceeds 5. The dimension of the word, field and position embeddings are set to 400, 50, and 5 respectively. Besides, the maximum position number is set to 30. Hence, any word in higher position number is considered to be at position 30. The most frequent 20k words and 1,480 fields in the training set are selected as word and field vocabulary respectively for both the encoder and the decoder.

4.3.2 Hyperparameters for Neural Question Generation

For the NQG task, we use a two-layer BiLSTM for the encoder and a single-layer LSTM for the decoder. We set the dimension of the LSTM hidden states to 350 and 512 for split-1 and split-2 respectively. Optimization is performed using AdaGrad optimizer with a learning rate of 0.3 and gradient clipping when its norm exceeds 5. The word embeddings are initialized with pre-trained 300-dimensional GloVe [52] which is freezed during training. We train the model up to 20 epochs with batch size of 50 and employ dropout with probability 0.1 and 0.3 for split-1 and split-2 respectively. Moreover, we use the vocabulary set released by Song et al. [60] for both the encoder and the decoder. During decoding, we perform beam search with beam size of 20 and length penalty weight of 1.75.

Models	BLEU-4	ROUGE-4
KN	2.21	0.38
Template KN	19.80	10.70
NLM [37]	4.17 ± 0.54	1.48 ± 0.23
Table NLM [37]	34.70 ± 0.36	25.80 ± 0.36
Table2Seq [5]	40.26	-
Order-planning [58]	43.91	37.15
Struct-aware Orig. [43]	44.89 ± 0.33	41.21 ± 0.25
Struct-aware Repl. [43]	44.45 ± 0.11	39.65 ± 0.10
Hierarchical Encoder + Auxiliary Supervision [42]	45.14 ± 0.34	41.26 ± 0.37
Our Model	46.07 ± 0.17	41.53 ± 0.30
+ EMA	**46.76 ± 0.03**	**43.54 ± 0.07**

Table 4.3: Experimental results for the neural table-to-text generation task on the test set. *KN* and *Template KN* are described in Section 2.3.

Models	Split-1			Split-2		
	BLEU-4	METEOR	ROUGE-L	BLEU-4	METEOR	ROUGE-L
PCFG-Trans [22]	-	-	-	9.47	31.68	18.97
s2s-ans [14]	12.28	16.62	39.75	-	-	-
SeqCopyNet [77]	-	-	-	13.02	-	44.0
NQG++ [76]	-	-	-	13.29	-	-
s2s+z+c+GAN [72]	-	-	-	13.36	17.70	40.42
MPQG [60]	13.98	18.77	42.72	13.91	-	-
s2s-a-at-mp-gsa [75]	15.32	19.29	43.91	15.82	19.67	44.24
AnsPos-aware [61]	-	-	-	15.64	-	-
RL_{ROUGE} [34]	16.17	19.85	43.90	-	-	-
ASs2s [29]	16.20 ± 0.32	19.92 ± 0.20	43.96 ± 0.25	16.17 ± 0.35	-	-
CGC-QG [41]	-	-	-	**17.55**	21.24	44.53
Our Model	14.81 ± 0.47	19.69 ± 0.24	43.01 ± 0.28	16.14 ± 0.25	20.44 ± 0.20	43.95 ± 0.26
+ EMA	**16.29 ± 0.04**	**20.70 ± 0.08**	**44.18 ± 0.15**	17.47 ± 0.10	**21.37 ± 0.06**	**45.18 ± 0.22**

Table 4.4: Experimental results for the NQG task on the test set.

4.4 Results

In what follows, we first present experimental results after training our model with MLE and then discuss the performance of the RL-refined model.

4.4.1 Evaluation after Maximum Likelihood Training

In Table 4.3 and 4.4, experimental results for neural table-to-text generation and NQG after training with MLE are listed. We conduct our experiments with three and five random seeds for neural table-to-text generation and NQG respectively, and report the mean and standard deviation of each evaluation metric. All results are copied from the original papers except for *Struct-aware* [43], in Table 4.3, for which *Repl.* refers to the scores from experiments that we conducted using the source code released by the authors, and *Orig.* refers to the scores taken from the paper. Additionally, we faced reproducibility issues for *ASs2s* when using the codebase released by the authors. Thus, we copied the results from the original paper. Moreover, *s2s-a-at-mp-gsa* has both paragraph-level and sentence-level versions. To ensure a fair comparison, we report their sentence-level results.

It is noteworthy that a similar version of our model has served as baseline in previous works [29, 41, 43]. However, the distinctions discussed in Section 2.3 and 2.4, especially adopting the EMA technique, enable our model to achieve state of the art in all cases except BLEU-4 on data split-2 of the SQuAD dataset for NQG in which the score is very competitive considering that Liu et al. [41] report a point estimate. This indicates two crucial facts. Firstly, a basic seq2seq model is able to effectively learn the underlying distribution of both datasets. Secondly, some previous works exhibit avoidable complexity in their models.

4.4.2 Evaluation after RL-refinement

In Table 4.5 and 4.6, we describe the results obtained from the model refined with the mixed objective (Equation 3.17) after 5000 iterations for neural table-to-text generation and NQG respectively. We repeat each experiment with three random seeds and report the mean and standard deviation of evaluation metrics for both tasks. In these experiments, we use BLEU, GLEU, and ROUGE as the reward function, and the mean of previous rewards as baseline (Equation 2.12).

Particularly, in Table 4.5, we observe that adding baseline almost always leads to better results when we use greedy search for generating Y sequences (see Equation 3.16). However, when we use multinomial sampling it results in better BLEU score but worse ROUGE score. In addition, for the case that we do not adopt baseline, multinomial sampling achieves much better results than greedy search. But, if we use baseline, greedy search slightly outperforms multinomial sampling.

Moreover, Table 4.6 shows that adding baseline when we use greedy search for generating output sequences leads to much better results for the case that ROUGE score is used as the reward function, but worse results for the case that either BLEU or GLEU score is used as the

Models	BLEU-4	ROUGE-4
Pre-trained model	**46.78**	43.47
+ RL_{BLEU}, w/ greedy search, w/o baseline	43.62 ± 0.13	43.45 ± 0.15
+ RL_{GLEU}, w/ greedy search, w/o baseline	43.30 ± 0.14	43.38 ± 0.35
+ RL_{ROUGE}, w/ greedy search, w/o baseline	41.70 ± 0.37	42.97 ± 0.07
+ RL_{BLEU}, w/ greedy search, w/ baseline	45.82 ± 0.07	43.26 ± 0.06
+ RL_{GLEU}, w/ greedy search, w/ baseline	45.65 ± 0.05	43.43 ± 0.09
+ RL_{ROUGE}, w/ greedy search, w/ baseline	45.61 ± 0.06	43.54 ± 0.04
+ RL_{BLEU}, w/ multinomial sampling, w/o baseline	45.78 ± 0.11	43.69 ± 0.06
+ RL_{GLEU}, w/ multinomial sampling, w/o baseline	45.40 ± 0.01	**43.84 ± 0.01**
+ RL_{ROUGE}, w/ multinomial sampling, w/o baseline	45.25 ± 0.09	43.56 ± 0.02
+ RL_{BLEU}, w/ multinomial sampling, w/ baseline	45.81 ± 0.13	42.95 ± 0.22
+ RL_{GLEU}, w/ multinomial sampling, w/ baseline	45.64 ± 0.23	43.20 ± 0.11
+ RL_{ROUGE}, w/ multinomial sampling, w/ baseline	45.48 ± 0.19	43.04 ± 0.22

Table 4.5: Experimental results after refining the model using RL with BLEU, GLEU, and ROUGE as the reward function for neural table-to-text generation.

reward function. Furthermore, incorporating baseline when we use multinomial sampling leads to better results specially when we use ROUGE and GLEU as the reward function. We also empirically observe that multinomial sampling achieves significantly better results than greedy search in NQG. We conjecture that this is mainly because using multinomial sampling results in better approximation of Equation 3.16.

Regarding NQG, we also use the F1 score of the BERT model for the generated questions as reward. Towards this, we first fine-tune the pre-trained BERT model on the training portion of data split-1 of the SQuAD dataset (see Section 4.1). We then utilize the F1 score of the fine-tuned BERT model to refine our pre-trained model for 2500 iterations. In Table 4.7, the obtained results for this experiment are reported. Inspired from Guo et al. [20], we also rescale the reward as we are not directly assessing the output of the generative model and to avoid vanishing gradient problem which is due to the small values of the reward function. The rescaling function is expressed below:

$$R_i = \sigma(\delta \cdot (0.5 - \frac{\text{rank}(i)}{B})) \tag{4.1}$$

where $\text{rank}(i)$ denotes the reward of the i-th element in the batch after sorting it from high to low, B is the batch size, δ is a hyperparameter that controls the smoothness of the rescale (set to 12), and σ is the sigmoid activation. This transformation ensures that the expectation and variance of

Models	BLEU-4	METEOR	ROUGE-L
Pre-trained model	16.29	20.76	44.12
+ RL_{BLEU}, w/ greedy search, w/o baseline	15.61 ± 0.10	20.51 ± 0.06	44.36 ± 0.07
+ RL_{GLEU}, w/ greedy search, w/o baseline	15.41 ± 0.09	20.45 ± 0.06	44.16 ± 0.08
+ RL_{ROUGE}, w/ greedy search, w/o baseline	14.00 ± 0.23	19.17 ± 0.26	43.12 ± 0.29
+ RL_{BLEU}, w/ greedy search, w/ baseline	15.15 ± 0.46	19.94 ± 0.33	43.16 ±0.6
+ RL_{GLEU}, w/ greedy search, w/ baseline	15.44 ± 0.31	20.13 ± 0.21	43.73 ±0.39
+ RL_{ROUGE}, w/ greedy search, w/ baseline	15.62 ± 0.08	20.26 ± 0.13	43.92 ± 0.33
+ RL_{BLEU}, w/ multinomial sampling, w/o baseline	16.43 ± 0.02	20.74 ± 0.05	44.67 ± 0.01
+ RL_{GLEU}, w/ multinomial sampling, w/o baseline	16.31 ± 0.07	20.72 ± 0.05	44.66 ± 0.06
+ RL_{ROUGE}, w/ multinomial sampling, w/o baseline	15.85 ± 0.13	20.50 ± 0.17	44.58 ± 0.07
+ RL_{BLEU}, w/ multinomial sampling, w/ baseline	16.45 ± 0.02	20.77 ± 0.02	44.69 ± 0.02
+ RL_{GLEU}, w/ multinomial sampling, w/ baseline	**16.48 ± 0.03**	**20.81 ± 0.01**	44.76 ± 0.01
+ RL_{ROUGE}, w/ multinomial sampling, w/ baseline	16.27 ± 0.04	20.72± 0.04	**44.91 ± 0.07**

Table 4.6: Experimental results after refining the model using RL with BLEU, GLEU, and ROUGE as the reward function on data split-1 of the SQuAD dataset for NQG task.

Models	BLEU-4	METEOR	ROUGE-L
Pre-trained model	16.29	20.76	44.12
+ RL_{BERT}, w/ greedy search	15.47 ± 0.18	20.48 ± 0.43	43.99 ± 0.17
+ RL_{BERT}, w/ multinomial sampling	15.49 ± 0.17	20.47 ± 0.12	43.99 ± 0.07
+ $RL_{BERT_{rescaled}}$, w/ greedy search	**16.38 ± 0.02**	20.75 ± 0.01	**44.30 ± 0.01**
+ $RL_{BERT_{rescaled}}$, w/ multinomial sampling	16.37 ± 0.03	**20.78 ± 0.01**	44.27 ± 0.02

Table 4.7: Experimental results after refining the model using RL with F1 score of BERT's output as the reward function on data split-1 of the SQuAD dataset for NQG task.

the reward in each batch are constant which is helpful to stabilize algorithms that are sensitive to numerical variance. As it is shown in Table 4.7, rescaling the reward has had a positive impact on the experimental results in comparison to the case that we directly use BERT's score without rescaling, for both greedy search and multinomial sampling. Besides, greedy search and multinomial sampling perform almost the same when we do not adopt rescaling.

4.5 Sample Outputs

Table 4.8 and 4.9 present several sample outputs for neural table-to-text generation and NQG respectively. In the second example of Table 4.8, birth date is mistakenly recorded on the birth place field in the dataset. Despite this error, our model is able to effectively use the date in the generated description. Interestingly, the model successfully converts the month number 8 to August, as in most ground-truth descriptions of the WIKIBIO dataset, the month is recorded by the name rather than the number.

Input table:	
name	dillon sheppard
full name	dillon douglas sheppard
height	1.80 m ftin 0 on
birth date	27 february 1979
birth place	durban, south africa
current club	bidvest wits
club number	29
position	left-winger
youth clubs	school of excellence
years	1999–2004 2004 2005 2006 2006–2009 2009–2011 2011–2013 2013–2014 2014–
clubs	school of excellence ajax cape town dynamo moscow panionios mamelodi sundowns platinum stars golden arrows maritzburg united
caps	24 105 4 10 37 39 33 14 0
goals	1 15 0 0 1 3 4 0 0
national years	1996 1997–1999 2000–2002 2000–2007
national team	south africa u-17 south africa u-20 south africa u-23 south africa
national caps	1 9 9 18
national goals	0 0 0 0
pcup date	23 august 2014
ntup date	2 june 2012
article title	dillon sheppard
Description:	
Ground-truth	dillon sheppard (born 27 february 1979) is a south african football (soccer) left-winger who plays for premier soccer league club bidvest wits and south africa .

Our model	dillon sheppard (born 27 february 1979 in durban , natal) is a south african football (soccer) midfielder who plays for bidvest wits in the premier soccer league .
Our model + RL_{BLEU}	dillon douglas sheppard (born 27 february 1979 in durban , south africa) is a south african footballer who plays as a left-winger for bidvest united f.c. in the premier soccer league .
Struct-aware	dillon sheppard (born 27 february 1979 in durban , kwazulu-natal) is a south african football (soccer) player who plays for bidvest wits .
Input table:	
name	narden jaime espinosa
order	member of the puerto rico house of representatives from the 35th district
term start	january 2 , 2009
predecessor	joel rosario hernández
birth place	1971 8 6
birth name	humacao , puerto rico
alma mater	university of turabo (b.a.)
children	2
party	popular democratic party (ppd)
article title	narden jaime espinosa
Description:	
Ground-truth	narden jaime espinosa (born august 6 , 1971) is a puerto rican politician affiliated with the popular democratic party (ppd) .
Our model	narden jaime espinosa (born august 6 , 1971) is a puerto rican politician affiliated with the popular democratic party (ppd) .
Our model + RL_{BLEU}	narden jaime espinosa (born march 6 , 1971) is a puerto rican politician affiliated with the popular democratic party (ppd) .
Struct-aware	narden , jaime espinosa is a puerto rican politician affiliated with the popular democratic party (ppd) .
Input table:	
name	kenny forssi
image	kenforssi.jpg
caption	ken forssi in 1980 in tampa , florida .
background	non vocal instrumentalist
birth name	kenneth raymond forssi
birth date	march 30 , 1943
birth place	cleveland , ohio , us
death date	5 january 1998

death place	tallahassee , florida
instrument	bass guitar
occupation	musician
years active	35
labels	elektra records , decca records
associated acts	love the surfaris
article title	ken forssi
Description:	
Ground-truth	kenneth raymond " ken " forssi (march 30 , 1943 – january 5 , 1998) , was an american musician , best known for being the original bass player in the band love .
Our model	kenneth raymond " kenny " forssi (march 30 , 1943 – january 5 , 1998) was an american musician , best known as the bassist for the band love the surfaris .
Our model + RL_{BLEU}	kenneth raymond " kenny " forssi (march 30 , 1943 – january 5 , 1998) was an american bass guitarist .
Struct-aware	kenneth raymond " ken " forssi (march 30 , 1943 – january 5 , 1998) was an american musician and composer .

Table 4.8: Sample outputs for neural table-to-text generation.

In the second example of Table 4.9, the question that is generated by *MPQG* model is basically addressing another answer *"there out of five"* which is mentioned before the actual answer *"one out of four"*. This could be due to the fact that *MPQG* does not employ answer position indicator and uses multi-perspective context matching algorithm [69] to match the passage and the target answer instead.

Passage:	the metropolitan area is also home to 20 % of the nation 's indian americans and at least 20 little india enclaves , as well as 15 % of all korean americans and four koreatowns ; the largest asian indian population in the western hemisphere ; the largest russian american , italian american , and african american populations ; the largest dominican american , puerto rican american , and south american and second - largest overall hispanic population in the united states , numbering 4.8 million ; and includes at least 6 established chinatowns within new york city alone , with the urban agglomeration comprising a population of 779,269 overseas chinese as of 2013 census estimates , the largest outside of asia .

Answer:	6
Question:	
Ground-truth	approximately how many chinatowns exist in new york city ?
Our model	how many established chinatowns are there in new york city ?
Our model + RL_{BLEU}	how many established chinatowns are within new york city alone ?
MPQG	how many chinatowns are there ?
Passage:	in 2005 , three out of five manhattan residents were college graduates , and one out of four had a postgraduate degree , forming one of the highest concentrations of highly educated people in any american city .
Answer:	one out of four
Question:	
Ground-truth	what fraction of manhattan residents have graduate degrees ?
Our model	how many manhattan residents had a postgraduate degree in 2005 ?
Our model + RL_{BLEU}	how many manhattan residents had a postgraduate degree in 2005 ?
MPQG	how many manhattan residents were college graduates in 2005 ?
Passage:	the largest reservoir in the state is fort peck reservoir on the missouri river , which is contained by the second largest earthen dam and largest hydraulically filled dam in the world .
Answer:	fort peck reservoir
Question:	
Ground-truth	what is the name of the largest reservoir in the state ?
Our model	what is the largest reservoir in the state ?
Our model + RL_{BLEU}	what is the largest reservoir in the state ?
MPQG	what is the largest reservoir in the missouri river ?

Table 4.9: Sample outputs for NQG.

Chapter 5

Conclusions and Future Work

In this thesis, we question the necessity of the existing complexity in NN architectures for text generation from structured (neural table-to-text generation task) and unstructured data (NQG task) and propose a simple, unified seq2seq model trained with EMA technique for both tasks. We use the WIKIBIO dataset for neural table-to-text generation and the SQuAD dataset for NQG. Empirically, our model is able to achieve state of the art in both tasks without using any linguistic features. This highlights the importance of exploring simple models before introducing novel, sophisticated neural architectures. We also investigate refining our model using reinforcement learning with different reward functions such as BLEU, GLEU, and ROUGE. Experimental results demonstrate that RL-refinement with these rewards barely improves the performance of our model. This could be because the model is well-trained after using the EMA technique during the maximum likelihood training phase. An interesting line of research for future work could be investigating the same scenario for transformers [67] by employing EMA technique in their training. Another potential direction for future work is distilling the knowledge [24] in recent large text generation models such as GPT-2 [54] into simple NN models.

References

[1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Rethinking complex neural network architectures for document classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4046–4051, 2019.

[2] Gabor Angeli, Percy Liang, and Dan Klein. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics, 2010.

[3] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017.

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

[5] Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. Table-to-text: Describing table region with natural language. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics, 2005.

[7] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[8] Samidh Chatterjee and Nicola Cancedda. Minimum error rate training by sampling the translation lattice. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics, 2010.

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

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[11] Matt Crane. Questionable answers in question answering research: Reproducibility and variability of published results. *Transactions of the Association of Computational Linguistics*, 6:241–252, 2018.

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*, 2017.

[15] Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, 2017.

[16] Pablo Duboue and Kathleen McKeown. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the International Natural Language Generation Conference*, pages 89–96, 2002.

[17] Yifan Gao, Jianan Wang, Lidong Bing, Irwin King, and Michael R Lyu. Difficulty controllable question generation for reading comprehension. *arXiv preprint arXiv:1807.03586*, 2018.

[18] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.

[19] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

[20] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H Clark, and Philipp Koehn. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 690–696, 2013.

[22] Michael Heilman. *Automatic Factual Question Generation from Text*. PhD thesis, Pittsburgh, PA, USA, 2011. AAI3528179.

[23] Michael Heilman and Noah A Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics, 2010.

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Tom Hosking and Sebastian Riedel. Evaluating rewards for question generation models. *arXiv preprint arXiv:1902.11049*, 2019.

[27] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[28] Joohyun Kim and Raymond J Mooney. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics, 2010.

[29] Yanghoon Kim, Hwanhee Lee, Joongbo Shin, and Kyomin Jung. Improving neural question generation using answer separation. *arXiv preprint arXiv:1809.02393*, 2018.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[32] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[33] Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346, 2013.

[34] Vishwajeet Kumar, Ganesh Ramakrishnan, and Yuan-Fang Li. A framework for automatic question generation from text using deep reinforcement learning. *arXiv preprint arXiv:1808.04961*, 2018.

[35] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

[36] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*, 2018.

[37] Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*, 2016.

[38] Percy Liang, Michael I Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics, 2009.

[39] Zachary C. Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. In *arXiv:1807.03341v2*, 2018.

[40] Bang Liu, Ting Zhang, Di Niu, Jinghong Lin, Kunfeng Lai, and Yu Xu. Matching long text documents via graph convolutional networks. *arXiv preprint arXiv:1802.07459*, 2018.

[41] Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. Learning to generate questions by learning what not to generate. *arXiv preprint arXiv:1902.10418*, 2019.

[42] Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6786–6793, Jul. 2019.

[43] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[44] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, 2017.

[45] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California, June 2016. Association for Computational Linguistics.

[46] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

[47] Ruslan Mitkov and Le An Ha. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, pages 17–22. Association for Computational Linguistics, 2003.

[48] Salman Mohammed, Peng Shi, and Jimmy Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks. *arXiv preprint arXiv:1712.01969*, 2017.

[49] Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059*, 2016.

[50] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[51] Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731, 2016.

[52] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[53] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[54] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8, 2019.

[55] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[56] David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

[57] Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257. Association for Computational Linguistics, 2010.

[58] Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. Order-planning neural text generation from structured data. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[59] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

[60] Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 569–574, 2018.

[61] Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, 2018.

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[63] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[64] Jun Suzuki and Masaaki Nagata. Cutting-off redundant repeating generations for neural abstractive summarization. *arXiv preprint arXiv:1701.00138*, 2016.

[65] Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*, 2017.

[66] Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1564–1574, 2018.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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

[69] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[70] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[71] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[72] Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. Teaching machines to ask questions. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4546–4552. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[73] Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*, 2017.

[74] Yuhao Zhang, Peng Qi, and Christopher D Manning. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, 2018.

[75] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, 2018.

[76] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer, 2017.

[77] Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. Sequential copying networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.