Question Generation with Adaptive Copying Neural Networks

by

Xinyuan Lu

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master in Computer Science With Specialization in Data Science

Ottawa-Carleton Institute for Computer Science
Department of Computer Science
Carleton University
Ottawa, Ontario

©2019
Xinyuan Lu
The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Question Generation with Adaptive Copying Neural Networks

submitted by Xinyuan Lu

in partial fulfillment of the requirements for the degree of

Master in Computer Science With Specialization in Data Science

________________________________________
Professor Yuhong Guo, Thesis Supervisor

________________________________________
Professor Diana Inkpen, Examiner

________________________________________
Professor Majid Komeili, Examiner

________________________________________
Professor Doron Nussbaum, Director
School of Computer Science

Ottawa-Carleton Institute for Computer Science
Department of Computer Science
Carleton University
September, 2019
Abstract

Given the rapid development of communication technology, online business websites are becoming increasingly popular. People could buy all the goods online rather than going to physical stores. However, it is often time-consuming for customers to read long product reviews when they make purchase decisions. Generally speaking, customers have specific questions in their minds when they seek information in product specifications. Therefore, if we can identify the most representative questions that can be answered by extensive product specifications, it will be much easier for customers to obtain their desired information quickly grasp, which is the motivation behind our thesis.

In this thesis, we aim to automatically generate questions from reviews. To tackle the automatic question generation (QG) task, we proposed a new adaptive copying recurrent neural network model, titled Adaptive Copying Neural Network (ACNN). The proposed model combines a component copying mechanism component onto a bidirectional LSTM structure to adaptively generate more suitable questions based on the input data. Subsequently, we calculated the generated questions’ summarization score to see if they could be answered by the reviews and to determine whether they were the most representative questions to summarize those reviews.

In our evaluation experiments, we first evaluated our ACNN model. We confirmed that our method can exceed the most advanced QG method in BLEU and ROUGE evaluation scores; it also showed better performance in human evaluation. In addition, we combined the summarization scores with our model which resulted in a preliminary performance boost. Overall, our model is fully data-driven and can simultaneously handle two significant tasks in natural language processing, which represents a new direction in this field.
Acknowledgments

Firstly, I would like to thank my supervisor, professor Yuhong Guo for supporting me with my research these two years. She is the one who help me get to where I am today.

Secondly, I would like to thank my friends in Machine Learning laboratory, Zengmao, Xuejun, Olivier. Thank you for answering all my questions. It’s really unforgettable to meet you guys. Especially, I would like to thank my best friend Siyu. Thank you for staying with me through the darkest time.

Last, I would like to thank my parents, who always support me and encourage me. Without their help, I could not insist on staying in Canada. You are my love! And my grandpa, you are always in my heart!!
# Table of Contents

Abstract iii  
Acknowledgments iv  
Table of Contents v  
List of Tables viii  
List of Figures ix  

1 Introduction 1  
1.1 Motivation 1  
1.2 Contributions 2  
1.3 Thesis Structure 3  

2 Background 4  
2.1 Introduction 4  
2.2 Deep Learning 4  
2.3 Neural Network 5  
2.3.1 Perceptron 6  
2.3.2 Feed Forward Neural Network 6  
2.3.3 Back Propagation 8  
2.4 Recurrent Neural Network 9  
2.4.1 LSTM 12  
2.4.2 GRU 14  
2.4.3 Bidirectional RNN 15  
2.5 Sequence-to-Sequence Learning Network 15  
2.5.1 Encoder-Decoder Network 16
2.6 Word Embedding ........................................... 19
  2.6.1 Glove Word Embedding .................................. 20
2.7 Mechanisms ................................................. 22
  2.7.1 Attention Mechanism ................................... 22
  2.7.2 Copy Mechanism ....................................... 24
2.8 Beam Search ................................................ 25
2.9 Perplexity ................................................... 25

3 Related Work ................................................ 27
  3.1 Introduction ................................................ 27
  3.2 Reading Comprehension ................................... 27
  3.3 Text Summarization ....................................... 29
  3.4 Review Summarization .................................... 31
  3.5 Question Generation ..................................... 31

4 Proposed Model ............................................. 34
  4.1 Goals .......................................................... 34
  4.2 Adaptive Copying Neural Network ......................... 36
    4.2.1 Encoder with Attention ................................ 37
    4.2.2 Decoder with Copy Mechanism ......................... 39
  4.3 Summary Score ............................................ 40
  4.4 Conclusion ................................................. 41

5 Experiments and Evaluation ............................... 42
  5.1 Introduction ................................................ 42
  5.2 Dataset ..................................................... 42
  5.3 Tokenization ............................................... 44
  5.4 Implementation Details ................................... 45
  5.5 Baselines ................................................... 46
  5.6 Evaluation Metrics ....................................... 48
    5.6.1 Automatic Evaluation Metrics ......................... 48
    5.6.2 Precision and Recall .................................. 48
    5.6.3 Human Evaluation ...................................... 49
  5.7 Evaluation Results ....................................... 50
    5.7.1 Automatic Evaluation Results for ACNN ............... 50
List of Tables

2.1 The co-occurrence matrix of the corpus ........................................ 20
5.1 Statistics of Du [1] dataset. ............................................................. 43
5.2 Statistics of Amazon dataset-Electronics. ......................................... 43
5.3 Statistics of Amazon dataset-Sports. ............................................... 43
5.4 Statistics of Amazon dataset-Tools and Home. ................................. 44
5.5 Statistics of question dataset for Laptop. ........................................ 44
5.6 Statistics of review dataset for Laptop. ........................................... 44
5.7 Human rating scheme in \textit{Correctness} ......................................... 49
5.8 Human rating scheme in \textit{Relevance} ........................................... 49
5.9 The comparison results in SQuAD. ............................................... 50
5.10 The comparison results in Electronics. ........................................... 50
5.11 The comparison results in Sports. ................................................ 50
5.12 The comparison results in Tools and Home. .................................... 51
5.13 Example of generated question and answer from Amazon datasets. ... 52
5.14 The results with different review lengths. ....................................... 53
5.15 Human evaluation results between our model and Du's model. ......... 53
5.16 Results of incorporating summary score in Laptop .......................... 54
5.17 Generated Questions by ACNN in Acer Review. .............................. 55
5.18 Generated Questions by ACNN+Repre in Acer Review. .................... 55
# List of Figures

2.1 A simple neural network [2]. ........................................ 5  
2.2 A simple structure of perceptron. .................................. 7  
2.3 A model of the a feed forward network perceptron. ............ 8  
2.4 A simple structure of a 2-layer feed forward neural network. . . 9  
2.5 A simple structure of a 3-layer feed forward neural network. ... 10  
2.6 A basic Recurrent Neural Network (RNN) model. .................. 12  
2.7 A cell of Long Short Term Memory network [3]. ................. 13  
2.8 A simple structure of Gated Recurrent Unit [4]. .................. 14  
2.9 A simple structure of Bidirectional RNN unfolded in three time steps [4]. 16  
2.10 A simple structure of a sequence-to-sequence learning network [5]. 17  
2.11 Recurrent Neural Network encoder-decoder [4]. .................. 18  
2.12 A simple example of word embedding [6]. ....................... 19  
2.13 A simple structure of a network with attention mechanism [32]. ... 23  
2.14 An example of dialog system [7]. ................................ 24  
2.15 The overall diagram of Copynet [7]. ............................. 25  
3.1 A network diagram for encoder [8]. ............................... 30  
4.1 The overall structure of our model. ............................... 36  
4.2 The structure of Adaptive Copying Neural Network. ............... 37  
5.1 An example of English word tokenization. ......................... 45  
5.2 Procedure of dropout [10]. ...................................... 47  
5.3 Sample output questions of our model and Du’s model. The copying word is highlighted in yellow. ......................... 57  
5.4 Example of an answer section of one Acer Laptop Review from Amazon. 58
Chapter 1

Introduction

1.1 Motivation

As a result of the fast development of electronic commerce, there are increasingly greater numbers of online websites. However, when people buy goods online, online product specifications are usually lengthy and take too long to read. It is a common mistake to think that product descriptions should simply convey the functionality of the product to the customer [11]. In fact, a broader conversion rate optimization strategy should include well-written, engaging, and convincing product descriptions [12]. It is worth pointing out that Google wants to rank high-quality product descriptions in search results, especially the uniqueness of product copies, which is easy to locate and understand when crawling websites [12]. In addition, when customers want to buy products online, most of them usually browse the instructions with particular questions in mind. If we were able to predict which product reviews will answer the questions that customers are eager to know, this would save customers a lot of time [13]. Therefore, this will enable customers to make purchase decisions in less time and help websites increase their sales.

Generally speaking, given a text span of specific length, the goal of question generation (QG) is to produce a pertinent readable question. However, human work is usually needed to solve the majority of QG problems. Handcrafted and well-designed rules are necessary when designing the questions. Humans then transform the input sentences into proper representations and generate questions using these perfect rules. However, we have used a fully data-driven methodology that only requires the paragraphs. We decided to use an Amazon’s question/answer
CHAPTER 1. INTRODUCTION

dataset based on Amazon products.

In addition, there are many applications for QG in people’s daily lives. For example, dialog systems in chatbots are currently drawing significant attention. Siri, Apple’s voice assistant, is a good application of chatbots that can talk with people as if they were human. Medical chatbots will soon function as assistants to diagnose patients’ symptoms. Moreover, QG is considered as a task related to question answering (QA). The goal of QA is to provide a reasonable answer to a given question. By forming questions, QG could boost the QA’s input and help improve both tasks.

Our approach contains two steps. Firstly, we applied the proposed Adaptive Copying Neural Network (ACNN) model to generate questions. Then, we identified the most representative questions given the specific document. To attain superior performance, we then improved our model through a copying mechanism.

1.2 Contributions

The thesis’ contributions (and their respective chapters) are as follows:

- We focus on extracting representative questions that are worth generating through the QG task, which is a previously ignored area. (Chapter 4)

- We propose a novel neural network ACNN by incorporating a copy mechanism component to tackle the QG problem. (Chapter 4)

- We design preliminary experiment and slightly boost performance by incorporating summarization scores in our ACNN model. (Chapter 4 and Chapter 5)

- We present the application of both summarization and the QG problem to Amazon product review data, which is useful for a variety of industries. (Chapter 5)
CHAPTER 1. INTRODUCTION

1.3 Thesis Structure

The rest of the thesis is organized as follows.

Chapter 2: We describe the concept of deep learning and the necessary background knowledge to understand our system. Readers familiar with deep learning and natural language processing may skip this chapter.

Chapter 3: We discuss several related works on previously proposed methods in QG, reading comprehension, review and text summarization.

Chapter 4: We illustrate the details of our system. We first define our task through formulas and then we describe our system based on two phases: the QG phase and the summarization score calculation. In the QG phase, we propose the ACNN model, which is a model to combine a copying mechanism component in a biLSTM network. In the second phase, we list our model’s formulas by incorporating a summary score.

Chapter 5: We provide the details of our experiment’s implementation. In addition, we introduce the study’s dataset, preprocessing step, and baseline. Finally, we describe the experiment’s evaluation metrics while explaining our results.

Chapter 6: This chapter contains the thesis’ conclusion and suggested future research in related areas.
Chapter 2

Background

2.1 Introduction

In this chapter, Section 2.2 and Section 2.3 describe the general concepts of deep learning and neural networks. Section 2.4 and Section 2.5 introduce the models commonly used in language processing and machine translation. Word embedding and Glove is explained in Section 2.6. Section 2.7 illustrates the mechanisms used in the deep network. Inference technique beam search is described in Section 2.8. At last, in Section 2.9, we introduce the evaluation method perplexity.

2.2 Deep Learning

During the last decades, given the development in various areas of computer engineering technology, deep learning has been greatly improved in complicated applications. In addition, larger amounts of training data in computer hardware as well as software—for example, multi-core GPUs—have nurtured the rise of deep learning.

Deep learning is an artificial intelligence technology that helps computers learn from experiences and the world around them. Deep learning is part of the machine learning approach, making data processing easier. However, unlike traditional statistic machine learning techniques, humans do not need to describe all the information a computer needs to run a task. Instead, computers can learn all the necessary knowledge through experience. Deep learning deals with problems by exploring the complex ideas that come from simple ideas. If the creation of these concepts is visualized, the process can be portrayed as a mind map—which is why it
is called “deep learning“.

2.3 Neural Network

In the human brain, neural networks consist of thousands of interconnected units called neurons. Neurons collect signals through dendrites and then collect signals through axonal outbursts, which divide into thousands of branches. At the end of each branch, synapses convert the activity of the axons into electrical effects, thereby stimulating further activity of the connected neurons [2]. A typical neural network model is shown in Figure 2.1.

Figure 2.1: A simple neural network [2].
2.3.1 Perceptron

The perceptron is an early ancestor of the neural network and the simplest type of neuron [14]. Each input has an associated weight, with a real number from 0 to 1 to denote the essence of the input towards the output. A perceptron’s output $A$ is defined as:

$$z = w \cdot x + b$$  \hspace{1cm} (2.1)

$$A = \begin{cases} 
0, & \text{if } z \leq 0 \\
1, & \text{if } z \geq 0
\end{cases}$$  \hspace{1cm} (2.2)

where $w$ and $x$ are vectors whose components are the weights and inputs separately.

$$w \cdot x = \sum_j w_j x_j$$  \hspace{1cm} (2.3)

Here, $b$ is a threshold value or bias. In addition, to convert a perceptron to a sigmoid neuron, the output is changed to $\sigma_g(z)$ which outputs a continuous value instead of a binary value. $\sigma_g(z)$ is the sigmoid activation function of $z$.

$$A = \sigma_g(z)$$  \hspace{1cm} (2.4)

$$\sigma_g(z) = \frac{1}{1 + e^{-z}}$$  \hspace{1cm} (2.5)

Figure 2.2 shows an example of a perceptron.

2.3.2 Feed Forward Neural Network

The feed forward neural network is a simple network in which each neuron in a layer is linked to each neuron input in the next layer. After taking a weighted linear sum over the inputs, an activation function is then applied. Figures 2.3 shows a simple model of a feed forward network perceptron. Figure 2.4 and Figure 2.5 show a 2-layer and a 3-layer feed forward neural network described within the book [15]
Figure 2.2: A simple structure of perceptron.

Hidden units are computed from the input units. The hidden layer $z$ in the $j^{th}$ layer is defined as

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + W_{j0}^{(1)}$$

$$z_j = h(a_j)$$

where $h$ is a nonlinear activation function such as sigmoid function. Each output is computed from the previous hidden layer. The variables $x$, $z$ and $y$ denote to inputs, hidden layers, and output, respectively. $M$ is the total output number and $D$ is the total input number.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

$$y_k = g(a_k)$$
2.3.3 Back Propagation

Back propagation is a common method of training neural networks by using optimization approaches like gradient descent (Algorithm 1). At the outputs, error function $\sigma$ needs to be calculated, which is the result of the activations from input in the forward pass. The error is then propagated backward. Next, the weights and the biases are adjusted with gradient descent. However, there are many other ways to perform back propagation, such as Direct Feedback Alignment [16], Target Propagation [17] and Particle Swarm Optimization [18]. In other words, back propagation is the most well-known way to train neural networks.

The gradient descent method is an iterative optimization algorithm that finds the minimum value of the objective function. Stochastic gradient descent (SGD) is a random approximation of gradient descent and is often used with backpropagation to train neural networks. However, one of the main problems with gradient descent is that it results in a local minimum value instead of a global minimum [19].
2.4 Recurrent Neural Network

Neural networks are a specific set of algorithms that revolutionize the field of machine learning. Their inspiration comes from biological neural networks, and deep neural networks have been evidenced to be effective [20]. The neural networks itself is general function approximations, which is the reason it can be employed for nearly all machine learning problem with complex input and output mapping [20].

The neural network model is a network in which many Logistics Units are organized according to different levels. The output variable of each layer is employed as the input variable of the next layer [21]. In essence, a neural network can obtain its own set of features through learning. In the general Logistic Regression, we were limited to using the original features \([x_1, ..., x_N]\) in the data. Although we can use some binomial terms to combine these features, we are still limited by these original features. In a neural network, the original feature is just the input layer. Like the example of the three-layer neural network in Figure 2.5, the third layer, the output layer, is made using the second layer of features instead of the input layer. In the original features, we can think of the features in the second layer as a series of new features that the neural network uses to predict the output variables after learning.
Kohonen’s self-organizing feature map (SOFM) is gaining more and more attention because SOFM has excellent features such as topology preservation and probability distribution retention. In numerous research, it is found that the predetermined network topology in SOFM implies the limitation of the result mapping. Usually, only after the training is finished, different network topologies may find better results [22]. This structured SOFM brings problems such as underutilization of neurons, inaccurate network mapping, and edge effects. In most cases, there is no prior knowledge that allows us to choose a suitable network size in advance. So it seriously affects the application of SOFM as the optimal vector quantizer.

Recurrent neural networks (RNNs) are commonly employed to process sequential data including text sequences, videos or speech. The output depends on the previous inputs [24]. RNNs have achieved good experimental results in machine translation, question generation and text summarization. There is a loop in the RNN architecture that computes the resulting memory up to the current time step. The typical architecture of RNN is illustrated within Figure 2.6.

**Figure 2.5:** A simple structure of a 3-layer feed forward neural network
Algorithm 1 The back-propagation training algorithm for feed-forward neural networks

function BackPropaganion (A,N):

Input: A training set A, a number of layers N

1: for each training example $x \in A$ do
2:  //Initialize the input layer.
3:  $a^{x,1} \leftarrow x$;
4:  // Forward pass.
5:  for each $n = 2, 3, ..., N$ do
6:     $z^{x,n} \leftarrow w^l \cdot a^{x,n-1} + b^n$
7:     $a^{x,n} \leftarrow \sigma_g(z^{x,n})$
8:     end for
9:  // Calculate the error.
10:  $\sigma^{x,N} \leftarrow \nabla_a C_x \circ \sigma'(z^{x,N})$
11:  // Backpropagate the error.
12:  for each $n = N, N-1, ..., 2$ do
13:     $\sigma^{x,N} \leftarrow ((w^{n+1})^T \sigma^{x,n+1}) \circ \sigma'(z^{x,N})$
14:     end for
15:  // Update weights and biases with gradient descent.
16:  for each $n = N, N-1, ..., 2$ do
17:     $w^n \leftarrow w^n - \frac{n}{m} \sum_x \sigma^{x,n}(a^{x,n-1})^T$
18:     $b^n \leftarrow b^n - \frac{n}{m} \sum_x \sigma^{x,n}$
19:     end for
20: end for

Let us take word sequence as an example. At each time step $t$, the input word $x_t$ and the previous hidden state $h_{t-1}$ are the input of the RNN. $x_t$ is the input word at time step $t$. The current hidden state is calculated as follows:

$$h_t = f(x_t, h_{t-1}) \quad (2.10)$$

where $f$ is a nonlinear function.

To calculate the probability, we used an encoder-decoder structure containing two recurrent neural networks. The target of the encoder is to convert a variable length input sequence into a background vector with fixed length. This background vector contains information about the input sequence. A commonly used encoder is a cyclic neural network [22]. A recurrent neural network can encode a tree/graph structure information into a vector, that is, map information into a semantic vector.
space [22]. This semantic vector space satisfies certain kinds of properties, such as semantically similar vectors that are closer together. That is to say, if two sentences (although the content is different) have similar meaning, then the distances of the two vectors encoded separately are similar. Conversely, if the meaning of the two sentences is completely different, then the distance of the encoded vector is very far [23]. The encoder network reads the input sequence and converts it into a vector representation. The decoder generates an output sentence based on these representations. In the next section, we will take a deeper look at our model structure.

### 2.4.1 LSTM

In the related natural language processing tasks, there are long-term dependencies problems that arise in sentence understanding. For example, given the sentence “I am from China, therefore I speak...“, the task is the prediction of the next word in the sentence. According to our common knowledge, the next word should be "Chinese" which is the name of the language. However, we can not predict it only by the previous word "speak". We need to go back to the beginning of the sentence and "China" is the most relevant information. This is the long-term dependency
problem. In order to solve this problem, a new network referred as Long and Short Term Memory (LSTM) is introduced below.

LSTM was first illustrated in [25] and it is a special type of RNN that can use gates to store knowledge that has already been shown for a long term. The LSTM network is different from traditional RNNs because LSTM has memory cells \( c_t \) in its hidden layer. Figure 2.7 demonstrates the architecture of a memory cell.

![Figure 2.7: A cell of Long Short Term Memory network [3].](image)

We employ the LSTM unit for the function \( f \). The activation function of the LSTM unit at time step \( t \) could be determined by the formula below:

\[
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)
\]

\[
f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)
\]
\begin{equation}
  c_t = f_t c_{t-1} + i_t \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c)
  \tag{2.13}
\end{equation}
\begin{equation}
  o_t = \sigma(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_{t-1} + b_o)
  \tag{2.14}
\end{equation}
\begin{equation}
  h_t = o_t \tanh(c_t)
  \tag{2.15}
\end{equation}

For each unit, \( \sigma \) denotes the logistic sigmoid function while \( i_t, o_t, f_t \) and \( c_t \) denote an input gate, an output gate, a forget gate and a cell memory, respectively. The weight matrix is \( W \), which varies within each equation. For example, \( W_{xf} \) referred to input-forget gate weight matrix while \( b \) is the bias \[3\].

Equation 2.11 and Equation 2.14 are the input gates and output gates that determine whether new information enters or exists. The forget gate in Equation 2.12 is used to determine what information should be kept or forget. In Equation 2.13, the cell memory \( c_t \) is updated at each time step with the previous cell memory \( c_{t-1} \) and the new information from input gate \[26\].

### 2.4.2 GRU

The Gated Recurrent Unit (GRU) network is another RNN to solve the long-term dependencies problem \[27\]. These networks are inspired by the LSTM unit. On the contrary, the structure is simpler. Figure 2.8 shows a simple GRU structure.

![Figure 2.8: A simple structure of Gated Recurrent Unit \[4\].]
Memory cells are not used by GRUs. They combine the input gate as well as forgetting gate into an update gate $z$ that determines the extent of the previous memory ought to be stored. GRUs also employ another gate referred as a reset gate ($r$) that determines how new inputs are combined with past memory. The hidden state at time $t$ is determined as below:

$$h_t = (1 - z_t) h_{t-1} + z_t s_t$$  \hfill (2.16)

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$  \hfill (2.17)

$$s_t = \tanh(W_s x_t + U_s (r_t \odot h_{t-1}))$$  \hfill (2.18)

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$  \hfill (2.19)

where $\odot$ is element-wise multiplication and $W_z$, $U_z$, $W_r$, $U_r$, $W_s$, $U_s$ are weight matrices which to be learned.

### 2.4.3 Bidirectional RNN

When considering the problems of forecasting a lost word within a sentence, the knowledge of the words located before and after the lost word is also important. To solve this problem, Bidirectional Recurrent Neural Networks (BiRNNs) [28] are illustrated here. Bidirectional RNNs are two different RNNs. One RNN reads the sentence from left to right, while the other RNN reads it from right to left. The output relies upon the hidden states of either RNNs. Then, concatenating the outputs of the forward hidden states, from left to right $\overrightarrow{b_j}$ as well as the output of the backward hidden states from right to left $\overleftarrow{b_j}$, by $b_j = [\overrightarrow{b_j}, \overleftarrow{b_j}]$ for input token $j$. Figure 2.9 shows the bidirectional RNN’s structure unfolded in three time steps [4].

### 2.5 Sequence-to-Sequence Learning Network

One of the constraints for deep learning network is that model’s input and output should be a fixed-length vector. However, in plenty of language processing tasks, the length of sentence is not fixed. In order to solve this problem, Sutskever [5]
developed a new technique called sequence-to-sequence learning model.

For example, in a task of machine translation, if we input ‘ABC’, we want to obtain ‘WXYZ’ as the output. The model is illustrated as follows (Figure 2.10) and it stops the output with the symbol “EOS” (end of sentence).

2.5.1 Encoder-Decoder Network

In order to complete the learning from sequence to sequence, Cho [4] proposed an encoder-decoder model, which also employed two different RNNs to process the data. The encoder and decoder can be RNNs, LSTM, or GRU. The encoder encodes the input sequence into a fixed length vector $c$, while in the decoder, a different length of sequence output is decoded by $c$.

The training goal of the whole model is to maximize the probability of output $y$ conditioned on input $x$. In the natural language processing tasks, the output could be various such as question in the QG task, the target language sentence in the
CHAPTER 2. BACKGROUND

Figure 2.10: A simple structure of a sequence-to-sequence learning network [5].

The encoder is used to read the sequence of input statements through the RNN, and the RNN continuously encodes the input until the input reaches the EOS token. The vector $c$ is then formed from a hidden state by a nonlinear function input sentence sequence through an RNN, which continuously encodes the input until it reaches the EOS token. Then, vector $c$ is generated from the hidden states by a non-linear function $q$ (Bahdanau [27]), which represented a summary of the statement of source. $T_x$ indicates the input sentence length.

$$X = (x_1, ..., x_{T_x})$$  \hspace{1cm} (2.21)

$$c = q(h_1, ... h_{T_x}) = h_{T_x}$$  \hspace{1cm} (2.22)
Decoder

The decoder is another RNN that is trained for the generation of output based on the previous hidden state and the summary vector $c$. The decoder’s hidden states are based on Equation 2.23 at time step $t$.

$$ s_t = f(s_{t-1}, y_{t-1}, c_t) \quad (2.23) $$

$s_{t-1}$ is the previous hidden state. The decoder decomposes the conditional probabilities into ordered conditionals in Equation 2.24.

$$ P(y|x) = \prod_{t=1}^{T} P(y_t|y_{t-1}, \ldots, y_1, x) \quad (2.24) $$

In addition, the conditional probability of determining the next output is calculated in Equation 2.25, where $g$ is an activation function. The softmax function is then employed for normalization:

$$ P(y_t|y_{t-1}, \ldots, y_1, x) = g(s_t, y_{t-1}, c_t) \quad (2.25) $$
2.6 Word Embedding

Word embedding, also referred as word vectors, comprises an N-dimensional vector of real numbers that is used to reduce the dimension from one per word in vocabulary size to N. Take an example of one hot word embedding. Suppose we have a vocabulary of five words: dog, cat, deer, book, bike. If we want to represent the words dog and bike, the vectors would be

dog = [1 0 0 0 0].
bike = [0 0 0 0 1].

A popular application of word embedding is to measure the words’ similarities. Specifically, if the two words have identical meanings, the embedding of the two words will be similar. In addition, if there is a specific connection among certain words, there will be a stable offset among the vectors. An example is illustrated within Figure 2.12.

\[ W(\text{“man”}) - W(\text{“woman”}) \approx W(\text{“king”}) - W(\text{“queen”}) \approx W(\text{“uncle”}) - W(\text{“aunt”}) \]

![Figure 2.12: A simple example of word embedding](image)

We can see that the male/female relationship is identical to the king/queen and uncle/aunt pairs. In natural language processing tasks, word embedding is a preprocessing step that could transfer words into vector representations. We can set
the word vectors randomly in advance or use several pre-trained word embedding algorithms. *Glove* word embedding is the pre-trained algorithm in our work which is provided by Pennington [29].

### 2.6.1 Glove Word Embedding

*Glove* [29], abbreviation of Global Vectors for Word Representation, is an algorithm used for word representations. The implementation of *Glove* is divided into the following three steps:

Firstly, it constructs a co-occurrence matrix $X$ based on the input corpus. The next example demonstrates the co-occurrence. Given three sentences in the corpus. "I like playing basketball" "I like books." "I enjoy eating.". The co-occurrence matrix is calculated in Table 2.1.

<table>
<thead>
<tr>
<th>counts</th>
<th>I like</th>
<th>enjoy</th>
<th>playing</th>
<th>basketball</th>
<th>books</th>
<th>eating</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>like</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>enjoy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>playing</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>basketball</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>books</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>eating</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Each element $X_{ij}$ in the matrix represents the number of times the word $i$ and the context word $j$ co-occur in a context window of a particular size. In general, the minimum number of co-occurrence is 1 but *Glove* [29] thinks otherwise. *Glove* proposed a decaying weight which is calculated in Equation 2.26.

$$ decay = \frac{1}{d} \quad (2.26) $$
This weight equation measures the distance of two words in a given context window $d$. The further away the distance is, the smaller the weight of the total count is.

Secondly, it constructs the approximate relationship between word vectors and co-occurrence matrix $[30]$. The authors of the paper proposed the following formula to approximate the relationship between word vectors and co-occurrence matrix.

$$w^T_i w_j + b_i + b_j = \log(X_{ij})$$  \hspace{1cm} (2.27)

Among them, $w^T_i$ and $w_j$ are the final word vectors we are looking for; $b_i$ and $b_j$ are the bias terms of the two word vectors respectively.

Thirdly, it constructs the loss function as shown in Equation 2.28 below.

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w^T_i w_j + b_i + b_j - \log X_{ij})^2$$  \hspace{1cm} (2.28)

The basic form of this loss function is the simplest mean square loss with an additional weight function $f(X_{ij})$ to it. There must be a significant amount of words in a corpus and they appear together frequently (frequent co-occurrence) $[30]$. Then we hope that: (1) The weight of these words is greater than those that rarely appear together (rare co-occurrences), so that function becomes non-decreasing; (2) The weight is not over-weighted and that it should plateau after reaching a certain level; (3) If two words do not appear together, that is $X_{ij} = 0$, as a result, they should not participate in the calculation of the loss function $[31]$, that is $f(x)$ has to satisfy $f(0) = 0$.

There are many functions that satisfy the above conditions, for example, a piecewise function is shown in Equation 2.29.

$$f(x) = \begin{cases} (x/x_{max})^a, & \text{if } x < x_{max} \\ 1, & \text{otherwise} \end{cases}$$  \hspace{1cm} (2.29)
2.7 Mechanisms

2.7.1 Attention Mechanism

Within the encoder-decoder model, we need to identify the input sentence’s fixed-length vector in advance. However, when input lengths increase, it becomes harder to calculate with vectors. In addition, Cho [4] showed the impact of the increased length of the input sentences when running the encoder-decoder model. Consequently, a long input makes it difficult to generate a good output.

To fix this problem, Bahdanau [32] proposed a model that can not constrain input to vectors with fixed length in machine translation tasks. This model combines alignment and translation of sentences. In the encoder step, it encodes the input sentence into a series of vectors. In the decoder step, it looks for the set of vectors to determine the best location (i.e., contains the most relevant information). Then, a context vector is formed based on the location. The model then predicts the next word based on these context vectors and previous generated words. The context vector $c$ involves summary of the input statements formed by the encoder as well as the hidden state of the encoder. The hidden state $h_i$ in encoder involves information concerning the whole input sentence with particular attention upon $i^{th}$ word. Context vector $c_i$ is determined as a weighted total of the encoder’s hidden states $h_i$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

(2.30)

The weight $\alpha_{ij}$ of each hidden state $h_j$ is calculated as follows:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

(2.31)

where $e_{ij}$ is a model of alignment that determines how much the inputs at position $j$ are matched with the output at position $i$. It is calculated in Equation
2.32. \[ e_{ij} = a(s_{i-1}, h_j) \] (2.32)

Alignment model $a$ can be a feedforward neural network. It is trained in conjunction with all other model parameters. Furthermore, $e_{ij}$ provides an attention mechanism within the decoder, which helps the decoder decide which part of the input sentence requires attention. As a result, the encoder does not need to compress all input information into a fixed-length vector. Bahdanau’s [32] attention model is illustrated within Figure 2.13.

**Figure 2.13:** A simple structure of a network with attention mechanism [32].

Luong [33] extended this network into local as well as global attention mechanisms. Within a global attention mechanism, the decoder focuses on all input words
while a local attention mechanism only focuses on one subset of input words at a time.

### 2.7.2 Copy Mechanism

A common phenomenon usually arises in the dialog system, that is, human tend to repeat special entity names and dates in both question and answer sentences. Figure 2.14 shows a typical example. I means input and R represents response.

| I: Hello Jack, my name is Chandralekha. | R: Nice to meet you, Chandralekha. |
| I: This new guy doesn’t perform exactly as we expected. | R: What do you mean by "doesn’t perform exactly as we expected"? |

We can see that the entity names are being copied from input to output. Therefore, Gu [7] proposed a copying mechanism to solve this problem. Basically, there are two questions to solve the copying problem: (1) Which sub-sequences should be copied and, (2) Where should that sub-sequence be pasted in the output.

Gu [7] proposed a model called *Copynet* which can be illustrated within Figure 2.15. We can see that the improvements basically happen in the decoder step. The first improvement is that they create both a generate mode and a copy mode to determine whether to copy or to generate the sub-sequence. The second improvement is that they calculate the location-specific hidden state in the decoder step which helps to focus on the location of information.

\[
    c^t = \sum_{t'=0}^{t-1} a^{t'}
\]  

(2.33)
In addition, the coverage vector is added to the attention mechanism as extra input. Therefore, it changes the attention mechanism formula to

\[ e^t_i = v^T \tanh(W_h h_i + W_s s_t + w_c c^t_i + b_{attn}) \]  \hspace{1cm} (2.34)

### 2.8 Beam Search

In natural language processing tasks, beam search (Algorithm 2) is a dynamic programming algorithm that is widely used to produce the output sentence. It is a breath-first-search algorithm. In each loop, the operations of adding and pruning the children are executed to create the final search tree. Finally, the search tree achieves a specific beam size number and stops when it reaches an EOS token.

### 2.9 Perplexity

In machine learning training process, the perplexity is an evaluation metric used to measure how well the model is. In QG tasks, the perplexity measures the performance of the output question \( Q \) based on the input answer \( A \). The perplexity is calculated
Algorithm 2. The beam search algorithm based on a heuristic and memory size.

function BeamSearch \((G, A, m, n)\):

**Input:** A graph \(G\), a rule set \(A\), a memory size \(m\), a goal node \(n\).

**Output:** A beam \(B\) of nodes starting at the root of \(G\) and ending on \(n\).

1. \(o \leftarrow\) new memory of size \(m\);
2. \(/ /\) Create a temporary node \(t\) and assign it the root of \(G\).
3. \(t \leftarrow G.Root();\)
4. \(o \leftarrow t;\)
5. \(B \leftarrow t;\)
6. while \(t \neq n\) do
7. \(\) Delete \(t\) from \(o;\)
8. \(\) Get children from \(t;\)
9. \(\) Delete children if they violate any rule \(\in A;\)
10. \(o \leftarrow\) children;
11. \(\) if \(o\) exceeds memory then
12. \(\) Delete worst node in \(o\) according to \(A;\)
13. \(\) end if
14. \(t \leftarrow\) best node in \(o\) according to \(A;\)
15. \(B \leftarrow t;\)
16. end while
17. return \(B\)

in Equation 2.35.

\[
\text{perp}(Q, A) = \exp\left(\frac{-\sum_{i=1}^{\left|Q\right|} \log P(q_i|q_{i-1}, \ldots, q_1, A)}{\left|Q\right|}\right)
\]  \hspace{1cm} (2.35)

\(q_i\) is the \(i^{th}\) output word in the question. To measure training perplexity, \(A\) and \(Q\) are taken from the training set. To measure validation perplexity, those words are taken from a validation set in epochs.
Chapter 3

Related Work

3.1 Introduction

This chapter provides an overview of recent work on QG and QA, reading comprehension, review summarization and text summarization problems. The chapter’s overall structure is as follows: Section 3.2 reviews reading comprehension problems and the methods to solve them. Section 3.3 and Section 3.4 describes the development of text summarization and product summarization methods. Finally, Section 3.5 reviews the QG problems.

3.2 Reading Comprehension

The modern era has witnessed exponential information growth. Reading all the generated texts is a time-consuming process. The main goal of reading generated texts is to understand those texts and then generate answers in response to questions. Thus, an automated understanding of machine reading is urgently needed.

Machine reading is defined by Anselmo Penas [34] as a task to handle automatic text understanding. The assessment of this understanding can be divided into two categories. The first method is to convert the text into a formal language representation followed by a query. This method is used for information retrieval. The second type is based on understanding natural language questions. The machine reading comprehension system is for understanding text content and related questions. Early reading comprehension systems used a simple method based on word package technology or manual defined rules. However, the level of understanding is
CHAPTER 3. RELATED WORK

not adequate for answering all input questions. For the provision of a better answer, researchers have created many ways to improve understanding.

Hirschman et al. [35] developed an automated system called DeepRead for reading comprehension. The system understood the passages by finding the answers to the user's questions from the original input. DeepRead used word package technology. After extracting the information content, the next task was to determine the greatest match between the question representations and the content.

Ellen Rilo and Michael Thelen [36] developed a rule on the basis of a QA system referred as Question Answering for Reading Comprehension, or QUARC. This system took the text and questions as input and then looked for the right answers to those questions. It employed part-of-speech tagging, morphological analysis, entity recognition, and semantic class tagging.

In 2000, Hwee Tou Ng [37] proposed a new QA system called the automatic question answering system, which used machine learning when reading stories. The system represented each question and answer pair as a feature vector, independent of the handwriting rules. This feature vector representation helped the learning algorithm automatically build five classifiers for each type of problem.

After the invention of the NLP tool, Simon Ostermann [38] proposed another system designed to correctly answer reading comprehension exercises for multiple-choice questions. The system is implemented in two steps. Within the first step, choose the sentence that best matches the question. Within the second step, the chosen sentences are compared with the four probable answers to calculate the similarity score. Preprocessing is done using NLP tools such as OpenNLP for sentence segmentation and Stanford CoreNLP for tagging.

Since 2015, many new reading comprehension datasets have been released. Hermann et al. [39] published a corpus in abstracts of CNN and Daily Mail news articles, which contained a large number of closed questions. Weston et al. [40] released bAbI, a fully synthetic dataset with 20 different tasks. Rajpurkar [41] released the Stanford Question Answering Dataset (SQuAD), a large QA dataset
that is commonly used for most reading comprehension tasks.

### 3.3 Text Summarization

Given the Internet’s dramatic growth, the amount of online information has expanded rapidly. Consequently, the demand for document summarization is also increasing quickly. Based on Radef et al. [42], a summary is less than half of the original text and also covers the text’s most essential information.

Automatic text summarization is another related task that automatically summarizes text while maintaining the essential meaning and keywords. Recently, numerous automatic text summarization approaches have been proposed and applied within various domains like search engines and online news topic headlines. Automatic text summarization can be truly challenging since it is difficult for computers to imitate human’s language ability and knowledge.

In general, there are two different approaches to automatic text summarization: extraction summarization and abstraction summarization. In extraction summarization, important words and sentences are identified in a text’s original content and then extracted from that text. This process depends on the original text and it functions like a highlighter. In contrast, abstraction summarization produces a summary in a new way by interpreting the original materials and then generating new words that can better describe the meaning. Thus, it functions more like a pen. Extractive summarization often provides superior results than abstractive summarization. This is because abstractive summarization always addresses problems such as semantic representation by learning the syntax of the words. Consequently, in today’s summarization systems, abstractive methods must still rely on extraction methods to some extent.

In 1995, Kupiec et al. [43] first described a traditional extractive method with machine learning. This method consisted of extracting the words in a document and then placing these inputs into Nave-Bayes classifiers, with a training set including long documents and the extractive summaries. The classification probability is acquired from the training data through Bayesian rules. The probability that a
sentence belongs to a digest is the score of the sentence. In addition to Nave-Bayes, machine learning classifiers includes decision trees, and conditional random fields are also common models for extracting abstracts.

As the fast progress of deep learning, plenty of new abstract generalization models have been developed in this field. In 2015, Rush et al. [8] first proposed an encoder-decoder, a full data-driven model named Attention-Based Summarization.

The network diagram for the encoder is shown in Figure 3.1. The aim is the maximization of the probability of $p(y_{i+1}|x, y_c; \theta)$. $enc$ stands for encoder model, which aims to incorporate the input sequence $x$ and context information. There are three types of encoder models in Rush et al. [8] study: convolutional encoder, bag-of-words encoder, and attention-based encoder. Figure 3.1(b) denotes an encoder based on attention.

![Network diagram](image)

**Figure 3.1:** A network diagram for encoder [8].

Nallapati et al. [44] improved the abstractive model through the following
modifications. First, they used the large vocabulary trick to solve the large decoder vocabulary problem. They incorporated linguistic characteristics such as TF-IDF, NER tag and POS tag to catch the sentences’ key parts. Finally, they also provided a new public dataset CNN/Daily Mail, for text summarization.

In 2017, Abigail See et al. [45] produced more recent work on abstractive summarization, proposing the Pointer-Generator for the solution of the problem of Out-Of-Vocabulary. Then, they employed a coverage mechanism to reduce the repeated vocabulary. They also conducted experiments on the CNN/Daily Mail dataset to achieve better results.

3.4 Review Summarization

Automatic review summarization is a hot topic of research in recent years. The purpose of review summarization is to integrate user input into a large overview of the product [46]. In terms of application tool, Yatani et al. [46] proposed an overview product review system called Review Spotlight. The system was based on extracting noun-adjective word pairs, then performed sentiment analysis of each word pairs, and showed the best word pair for the product review. Ly et al. [47] developed a review summarization system that consisted of product facet identification and subsequent summarization. In 2016, Liu et al. [48] first summarized the review with information retrieval and then extended aspects of the review by sentence ranking. In our work, the goal is similar to the review summary that aims to shorten the product review. But the difference to existing product review summarization is that we are going to choose the representative questions to express the major points of discussion within the review.

3.5 Question Generation

In general, methods for solving QG problems are divided into four categories: syntax-based, semantic-based, template-based and sequence-to-sequence learning models. The syntax-based model is better for shorter sentences; however, it can sometimes produce grammatical mistakes. Semantic-based methods apply semantic role labeling techniques to identify targets. Template-based approaches are more
efficient in purpose-specific domains where human resources are usually needed to create templates. Sequence-to-sequence learning model, which are not dependent on human rules is more effective for solving QG problems.

Chen et al. [49] expanded their original work on auto-generating problems from a narrative text. In the later work, Chen et al. [49] mainly summarized their previous information text technology to help children in the first to third grade have a clearer understanding of the text.

Kalady et al. [50] proposed a grammar-based QG problem solving method, which used key word modelling. Researchers generated facts and definition questions by employing named entity recognition. Therefore, they simultaneously produced yes/no and wh type questions.

In 2010, Liu et al. [51] proposed a system that generated triggering questions to support students automatically writing literature reviews. In this process, the citation and core content elements can first be extracted from the student’s writing. Second, by using rule-based techniques, they classified references and generated problems based on a series of templates. Heilman and Smith [52] conducted another study in 2010 using a template-based approach. Their tricks include generating question and then sorting them. In order to convert the declarative sentences into questions, they developed a set of handwritten rules to syntactically convert the sentences. In addition, they used a logistic regression model for ranking.

With another work in 2010, Heilman and Smith [52] solved the QG problem, which is factual information about reading materials with complex grammar. Through their strategy, they proposed an algorithm that extracted simple and accurate fact statements from a semantic and syntactic perspective. They proved that their system was more practical than the standard text compression algorithm that handled factual problems.

In view of the successful development of method of deep learning in natural language processing tasks, QG has also developed rapidly. Serban et al. [53] conducted a study on the use of neural networks in QG mission in 2016. The authors
solved the problem by turning knowledge map facts into questions. Subsequently, they promoted a factoid question and answer corpus through the RNN architecture.

QG can also be combined with its complementary task QA. Tang et al. [54] considered QG as well as QA as dual tasks while training the model. The researchers’ training framework took advantage of the probabilistic correlation between the two missions. Song et al. [55] also developed a model to solve these two problems. They applied an attention-based encoder-decoder model that relied on the targeted answer as well as the article as input. The model then performed query comprehension to capture more interrelationships between the articles and target answers.

Du et al. [1] in 2017 conducted a supplementary study. The researchers used a sequence to sequence learning model to generate questions from sentences as well as paragraphs. Similarly, Zhou et al. [11] used the same sequence to sequence learning model with the different input answer NER tags.

Yuan et al. [13] also contributed to this field in 2017 with a relevant study. The authors utilized unlabeled text to boost the performance of QA models. They used semi-supervised learning and reinforcement learning in their algorithms in order to alleviate the discrepancy between model-generated and human-generated data distribution.

In 2018, Gao et al. [56] did research in the difficulty level of QG dataset SQuAD. They also proposed a novel question generation setting called Difficulty-controllable Question. By controlling the different type of questions with a difficulty label, they found that the performance are better comparing to without it.
Chapter 4

Proposed Model

4.1 Goals

Traditional Internet search engines use keywords as a medium to handle users' information needs. In general, feedback from users is a hyperlink to massive web pages [11]. In order to alleviate the pressure of users to re-screen these information feedback, the research on the method of sorting the retrieved web pages has received great attention. However, this method of information retrieval with keywords as input and output by sorted webpage hyperlinks does not substantially improve the effect of human information acquisition [12]. Recent research shows that human information needs are usually expressed in the form of natural language problems, and the expectations is an accurate answer towards the questions mentioned [12]. Therefore, the way of information acquisition in the future will be oriented towards the direction of the question and answer system.

Question Generation is an emerging field of research. The starting point of this research is that it hopes to help people improve their information needs by automatically generating a large number of problems with good linguistic features [13]. Since 2008, an international seminar was held every year to facilitate researchers from all over the world to exchange the latest research results, including international and regional researchers such as question and answer system, dialogue generation, and intelligent teaching (Germany, India, Canada, USA, UK, etc.) [1]. And at the 2010 seminar, the evaluation of automatic question generation was held for the first time. The evaluation is mainly on sentence-based question generation and paragraph-based question generation based on the corpus size of the question [1].
CHAPTER 4. PROPOSED MODEL

The RNN model using the traditional encoder-decoder first learns the input sequence with some LSTM units, encoding it as a fixed-length vector representation; then uses some LSTM units to read this vector representation and decode it into an output sequence [1]. Models with this structure have achieved the best results in many difficult sequence prediction problems (such as text translation) and have quickly become the current mainstream method [57]. Overall, most of the current research focuses on sentence-based problem generation, and only the University of Pennsylvania is involved in paragraph-level problem-solving evaluations. The mainstream method of sentence-based problem generation is to adjust the word order to generate corresponding problems according to the pre-established rules after syntactic analysis and named entity recognition. Although the LTI Center of Carnegie Mellon University proposed in 2009 to increase the diversity of problems by over-producing problems and using statistical methods to score a number of candidate questions, the core of the problem is still the use of rules [58].

As the demand for problem shows, researchers hope to extract sentences and even questions from paragraphs directly from the document. Some researchers believe that the sentence and paragraph in the document should be treated differently. The sentence level question generation selects the candidate sentences into a fixed question set through the mapping between the pre-defined syntax template and the question component. Then, the word order is adjusted according to different question types to achieve the purpose of the question generation [58]. For the question at the paragraph level, it is to filter the keywords and used the fixed template to generate the definition based on the semantic type of the keyword [57]. Some scholars believe that relying solely on syntactic analysis of such shallow language analysis is not sufficient to achieve the quality criteria for automatically generating questions, especially if the questions sometimes generated by syntactic templates alone are often unreadable [58]. Therefore, the researchers hope to determine the type and form of the question by finding the semantic relationship between the components in the sentence. At present, some scholars propose the subject-object relationship block with verb as the core base for generating the question, and simultaneously analyze the semantic type of the component as the type of auxiliary problem.

Our model has two goals: to generate questions automatically and to select
the representative questions to summarize a review. First, we will define the goal of the QG mission. Given an answer to a sentence or a paragraph $x$, it is a sequence of tokens $[x_1, ..., x_N]$ with a maximum length of the answer $N$, we aim to generate a natural question $y = [y_1, ..., y|Y|]$ with the length of $Y$ from the answers. In other words, our goal is to maximize the conditional probability $p(Y|X)$. Second, we calculate the summary score to measure the link between the answer section and the questions; this process is detailed in Section 4.3. Figure 4.1 shows the overall structure of our model.

4.2 Adaptive Copying Neural Network

In this thesis, we proposed a novel Adaptive Copying Neural Network which we used for the SIGMOD Student Research Competition [9]. We used a bidirectional LSTM with a global attention mechanism to perform automatic QG. In addition, we added a copying mechanism component to this neural model that incorporate the original input vocabulary information into the decoding phase to generate the appropriate questions. The proposed ACNN model is shown in Figure 4.2. This end-to-end learning model has two basic parts, an attention-based encoder and a
copying mechanism-based decoder.

![Image](image.png)

**Figure 4.2:** The structure of Adaptive Copying Neural Network.

### 4.2.1 Encoder with Attention

The most basic seq2seq model consists of an encoder and a decoder. Typically, the input statement encodes a state of fixed size followed by as the original state of the decoder (of course, as an input for each moment). But such a state is the same for all moments in the decoder. The copy mechanism is added to the LSTM to describe the new object, and the probability of getting the next word using the decoding generation and copy mechanism while predicting the next word in the LSTM decoding process [59]. The words obtained by the copy mechanism may not appear in the paired data set that LSTM uses for training, so that a new mechanism outside the paired data set can be described using a copy mechanism [60]. The Attention-based model is actually a measure of similarity. The more similar the current input is to the target state, the more the current output depends on the current input [60]. We can say that attention is not a
new model, but based on previous patterns to increase the concept of attention. Thus, one cause or attention mechanism model is a reasonable name, not a model [60].

As for encoder, we used an attention based bidirectional LSTM of the given token sequence within the input sentence $x$.

Let $\overrightarrow{h_t}$ denotes the hidden state at time step $t$ for the forward LSTM and $\overleftarrow{h_t}$ denotes the backward LSTM. The bidirectional LSTM produces the hidden states as follows:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(x_t, \overrightarrow{h_{t-1}}) \quad (4.1)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(x_t, \overleftarrow{h_{t+1}}) \quad (4.2)$$

We then concatenate the hidden states at time step $t$ from both directions to obtain the final hidden state of encoder where $;$ represents the concatenation operation.

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \quad (4.3)$$

The attention-based encoding of $x$ at a decoding step $k$ is then computed as a weighted average of the representation vectors across $h_t$,

$$c_k = \sum_{t=1}^{N} a_{k,t} h_t \quad (4.4)$$

The attention weights $\{a_{k,t}\}$ are then calculated using a softmax normalization.

$$a_{k,t} = \frac{exp(e_{k,t})}{\sum_{j=1}^{N} exp(e_{k,j})} \quad (4.5)$$

$$e_{k,t} = tanh(d_k^T W_h h_t) \quad (4.6)$$

where $W_h$ is the model parameter to be learned and $d_k$ is the hidden decoding state at time step $k$ which we will introduce below.

We also consider using the bidirectional encoding LSTM to replace the encoding of $x$, encoding the truncated paragraph (with length $L$) containing the sentence $x$. 
4.2.2 Decoder with Copy Mechanism

At present, the deep learning-based language generation model is mostly based on the original data learning numerical calculation model. How to integrate the external knowledge base of symbol representation in the natural answer generation process is a big challenge \cite{61}. In addition, many questions need to use multiple facts in the knowledge base, and different semantic units (words, entities) of a natural answer may need to be obtained through different ways. Answering this complex question that requires multiple patterns to extract and predict semantic units poses a greater challenge to the generation of natural answers \cite{61}.

The decoding process here is to generate the question $y$ from the given sentence $x$, which is a probability sequence prediction that can be decomposed into:

$$P(y|x) = \prod_{k=1}^{|y|} P(y_k|y < k, x)$$

(4.7)

This is also the probability of experience that we need to maximize in the training process of all the annotated training samples.

By integrating both an LSTM attention-based decoding component and a copying mechanism component developed in \cite{1}, the local conditional word output probability $P(y_k|y < k, x)$ is calculated, ie

$$P(y_k|y < k, x) = z_k P_{cop}(y_k) + (1 - z_k) P_{att}(y_k)$$

(4.8)

The attention part $P_{att}(y_k)$ generates words from the common decoder vocabulary, and it is computed on the attention vector $c_k$ and the hidden state vector $d_k$ from an LSTM decoding:

$$P_{att}(y_k) = \text{softmax}(W_y \tanh(W_k[d_k; c_k]))$$

(4.9)

where $W_y$ and $W_k$ are model parameters. The hidden decoding state $d_k$ at time step $k$ which is a LSTM is generated by:

$$d_k = LSTM(y_{k-1}, d_{k-1})$$

(4.10)
where $y_{k-1}$ is the previously generated word.

The copying mechanism component $P_{\text{cop}}(y_k)$ generates (copies) words from the individual vocabulary of the source input sentence. We compute it as

$$P_{\text{cop}}(y_k) = \text{softmax}(V^T(V[d_k; c_k] + b_1) + b_2) \quad (4.11)$$

where $y_k$ derives from the unique word set of the source input sequence, where $V$, $b_1$ and $b_2$ are model parameters. Such a copying mechanism can help incorporate words from the original data into the generated questions.

The combination weight $z_k$ is the switch that decides whether to generate a word from the vocabulary or to copy it from the input sequence. Inspired by [45], we calculated $z_k$ as follows:

$$z_k = \sigma(W_d^T d_k + W_c^T c_k + W_s^T y_{t-1} + b) \quad (4.12)$$

where $W_d$, $W_c$, $W_s$ are model parameters, and $\sigma$ denotes a sigmoid function. Hence, $z_k$ functions as a selection gate that makes data adaptive selection between the attention component and the copying component.

### 4.3 Summary Score

To extend the work of Liu et al. [48], if we want to represent a product review, we must measure the connections between the question and the review. Product review refers to the quality audit activity carried out in order to obtain the complete quality information of the product. It is a management tool for independent review of products from the customer’s point of view [62]. In other words, we must assess whether these questions could be adequately answered through the review. Therefore, we decided to calculate the score between review $x$ and question $y$ by adding the answer section $s$:

$$\text{summary score}(y, x) = \sum_s P((y, s)|x) = \sum_s P(y|s)P(s|x) \quad (4.13)$$
where \( s \) represents a part of the review and \( P(s|x) \) represents the importance of \( s \) in review. Then, \( P(s|x) \) can be estimated by the percentage of the number of related words in the answer sections and the reviews respectively. This value can be calculated by the following equation, and we assume that the question is only dependent on the answer section \( s \).

\[
P(s|x) = \frac{\text{number of words}(s)}{\text{number of words}(x)}
\]

(4.14)

\( P(y|s) \) denotes the question with the largest probability given an answer section \( s \). It could be calculated by Equation in our ACNN model.

### 4.4 Conclusion

Within this chapter, we discuss the motivation behind our system as well as described our problem and proposed model with formulas. We have adopted a global attention mechanism and the copy mechanism into our ACNN model. Within is chapter, we also demonstrate the summary scores for selecting representative questions. In the next chapter, our experiments and model evaluation methods will be introduced.
Chapter 5

Experiments and Evaluation

5.1 Introduction

This chapter presents the implementation details of proposed ACNN model (Section 5.3 and Section 5.4), including the dataset’s details (Section 5.2). We then describe the study’s evaluation metrics for testing the experimental results, including three automatic evaluation metrics and the human evaluation metric in Section 5.6. At last, Section 5.7 provides an analysis of our model’s results with baselines.

5.2 Dataset

Throughout this thesis, we used SQuAD and Amazon Question/Answer as our datasets for comparison. For the SQuAD dataset, we used the version released by Du et al. [1] in 2017, which was divided into three parts: development set and training set as well as the test set. This dataset involves 536 articles with over 100,000 questions regarding those articles. The researchers randomly split 80% of the dataset into a set of training, 10% into a development dataset and 10% into a test dataset. Here, we report our experimental results using the test dataset. Table 5.1 shows the dataset’s statistics.

We also used the Amazon Question/Answer dataset released by Wan and McAuley [63] in 2016. This dataset contains 21 total product categories. We focused

---

1http://www.amazon.com
2http://jmcauley.ucsd.edu/data/amazon/qa/

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of pairs (Training set)</td>
<td>70,484</td>
</tr>
<tr>
<td>No. of pairs (Development set)</td>
<td>10,570</td>
</tr>
<tr>
<td>No. of pairs (Test set)</td>
<td>11,877</td>
</tr>
<tr>
<td>Sentence average tokens</td>
<td>32.9</td>
</tr>
<tr>
<td>Questions average tokens</td>
<td>11.3</td>
</tr>
</tbody>
</table>

on three product categories: Electronics, Sports, and Tools and Home.

These categories’ statistics are listed within Table 5.2, Table 5.3 and Table 5.4, respectively. We split 80% of the dataset into a training set, 10% into a development dataset and 10% into a test dataset.

Table 5.2: Statistics of Amazon dataset-Electronics.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of pairs (Training set)</td>
<td>93,527.2</td>
</tr>
<tr>
<td>No. of pairs (Development set)</td>
<td>11,690.9</td>
</tr>
<tr>
<td>No. of pairs (Test set)</td>
<td>12,101</td>
</tr>
<tr>
<td>Sentence average tokens</td>
<td>20.5</td>
</tr>
<tr>
<td>Questions average tokens</td>
<td>9.7</td>
</tr>
</tbody>
</table>

Table 5.3: Statistics of Amazon dataset-Sports.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of pairs (Training set)</td>
<td>37,645.6</td>
</tr>
<tr>
<td>No. of pairs (Development set)</td>
<td>4,705.7</td>
</tr>
<tr>
<td>No. of pairs (Test set)</td>
<td>4,897</td>
</tr>
<tr>
<td>Sentence average tokens</td>
<td>20.5</td>
</tr>
<tr>
<td>Questions average tokens</td>
<td>9.6</td>
</tr>
</tbody>
</table>
Table 5.4: Statistics of Amazon dataset-Tools and Home.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of pairs (Training set)</td>
<td>30,136.2</td>
</tr>
<tr>
<td>No. of pairs (Development set)</td>
<td>3,767</td>
</tr>
<tr>
<td>No. of pairs (Test set)</td>
<td>3,910</td>
</tr>
<tr>
<td>Sentence average tokens</td>
<td>20.6</td>
</tr>
<tr>
<td>Questions average tokens</td>
<td>9.9</td>
</tr>
</tbody>
</table>

To evaluate the impact of summary score, since it is a preliminary experiment and has the large amount of work, we chose to focus on 50 products in the Laptop category. The laptop category’s overall statistics are featured in Table 5.5. We then selected five reviews for each product. In addition, our minimum review length is 150 words. We first separate each answer into two or three answer sections. Then, we pair each review section with questions. The answer section data’s statistics are shown in Table 5.6.

Table 5.5: Statistics of question dataset for Laptop.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of products</td>
<td>331</td>
</tr>
<tr>
<td>No. of questions</td>
<td>8,781</td>
</tr>
<tr>
<td>Questions average tokens</td>
<td>10.8</td>
</tr>
</tbody>
</table>

Table 5.6: Statistics of review dataset for Laptop.

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Number of Review Sections</td>
<td>12.57</td>
</tr>
<tr>
<td>Average tokens per section</td>
<td>50.334</td>
</tr>
</tbody>
</table>

5.3 Tokenization

Tokenization is a text data preprocessing step. The goal of tokenization is to split long texts into smaller chunks that can be a word or a single sentence. These chunks are called tokens. Figure 5.1 is an example of word tokenization from Stanford...
In the English word tokenization, the tokens are naturally split by spaces between the original words. In contrast to English, Chinese writing has no blank spaces between each single character. As a result, the punctuation between the Chinese sentences is an obvious demonstration used for Chinese sentence tokenization.

In our work, we use NLTK toolkit to perform the word tokenization. We also assigned an index to each of these words to make them different. Moreover, we abandoned the questions that have less than three words.

5.4 Implementation Details

We implemented our model using the PyTorch package integrated in Python. PyTorch is a deep-learning framework, providing maximum flexibility and speed. It is calculated by the neural network and the depth of the tensor computed by the n-dimensional array GPU composition. Tensor can be placed on the CPU or GPU. PyTorch pack more efficient use of memory than other tools.

The first time before training is to build our source and target vocabulary. To do this, we find all the unique words on the source and target sides and assign an index to each word. We set the source vocabulary size to 45k and the target vocabulary size to 28k. Only the most frequent words were stored in each vocabulary, and the rest of the words are marked with UNK token. Other tokens BOS and EOS indicate the beginning and end of the sentence, respectively. We set 300 as the embedding word vector dimension and used glove.840B.300d released by Stanford NLP Group as pre-training word embedding. 840 billion tokens are collected from

---

2http://jmcauley.ucsd.edu/data/amazon/qa/
3http://www.nltk.org/
4http://torch.ch/
5https://nlp.stanford.edu/projects/glove/
web data. If any word vocabulary does not exist in the *glove* word embedding, the random number is used to initialize the weight. Word vectors are updated during training process and finally are fine-tuned for our tasks.

In the model structure, we set the size of a hidden state unit to 600 as well as the layer number to 2 followed by initializing the model parameters in a uniform distribution. The stochastic gradient descent method (SGD) was used as optimization approach, and the initial learning speed was 1.0. It is a useful method when dealing with very large datasets. This method uses only a few training samples or small batches instead of an example to update the model parameters. The learning speed represents the speed at which we use to update the model parameters. After 8 epochs and the end of each remaining epoch, we began to halve the learning speed. And we continued to train 20 epochs. When the model tries all of the datasets samples, one epoch is completed. We set the batch size to 64, which means that sample number passed forward or backward is 64.

In addition, we set the probability of dropout to 0.3. Dropout [10] is a regularization technique employed to prevent overfitting problems by randomly deleting cells from the network during training. Figure 5.2 illustrates the dropout process in a neural network.

Decoding is performed using a beam search with a size of 5, which stops when we encounter an EOS token. Finally, we chose the model with the lowest perplexity on the validation set, which indicates how well the model matches the data.

### 5.5 Baselines

To evaluate our ACNN model in the QG step, we compared it with the model provided by Du et al. [1], shortened to *Du* in the following section. Du et al. [1] applied an attentional LSTM model that works on sentence and paragraph levels and aims to complete the task of QG. Du et al. [1] tested the model using the Stanford Question Answering Dataset. We only select their sentence-level as a baseline model.
and conducted additional experiments using the Amazon Question/Answer dataset.

To evaluate the summarization score, we only did preliminary experiment on the *Precision*, *Recall* and *F*₁ rate in the ACNN model for comparison.
5.6 Evaluation Metrics

5.6.1 Automatic Evaluation Metrics

For the evaluation of our ACNN model’s influence, we used three automatic evaluation metrics, namely BLEU-1, BLEU-2 and ROUGE-L. The package that we applied for these metrics is provided by Chen et al. [64]. The sections below introduce how to calculate those metrics.

**BLEU**

BLEU-\(n\) (Bilingual Evaluation Understudy) [65] is an \(n\)-gram score employed to map the output generated by the computing machine to the true value of the ground truth. Within our work, we just considered BLEU-1 as well as BLEU-2, which calculate the overlap between one-gram and bigrams.

**ROUGE**

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [66] measures the co-occurrences of the longest common subsequence. Equation 5.1 is to calculate the ROUGE-L score.

\[
ROUGE_L = \frac{LCS(output, ground truth)}{Total \ number \ of \ words \ of \ ground \ truth} \tag{5.1}
\]

In general, the higher scores in these two metrics indicate better performance.

5.6.2 Precision and Recall

To evaluate the impact of the model’s summarization score, we chose to use the precision and recall evaluation metrics, which are commonly used in information retrieval. Precision, recall, and F1 are calculated by the following equations.

\[
Precision = \frac{number \ of \ overlapping \ words}{Total \ number \ of \ words \ in \ system \ summary} \tag{5.2}
\]

\[
Recall = \frac{number \ of \ overlapping \ words}{Total \ number \ of \ words \ in \ reference \ summary} \tag{5.3}
\]

\[
F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.4}
\]
5.6.3 Human Evaluation

To better analyze our model’s performance, we conducted human evaluations on the results of ACNN model and Du’s model. The criteria included Correctness and Relevance. The Correctness criterion judges the grammar and fluency of the generated questions, while Relevance indicates whether the answer sentence is relevant to the generated questions. The output score ranges from 1 (very poor) to 5 (excellent). Table 5.7 and Table 5.8 showed the rating scheme of human evaluation.

Table 5.7: Human rating scheme in Correctness

<table>
<thead>
<tr>
<th>Score</th>
<th>Rating Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5: Excellent</td>
<td>This question is perfectly in grammar and easy to read.</td>
</tr>
<tr>
<td>3: Neutral</td>
<td>Some grammar mistakes in the question and I need to guess the question meaning.</td>
</tr>
<tr>
<td>1: Very Poor</td>
<td>I can not totally understand this question.</td>
</tr>
</tbody>
</table>

Table 5.8: Human rating scheme in Relevance

<table>
<thead>
<tr>
<th>Score</th>
<th>Rating Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5: Excellent</td>
<td>This question is meaningful and matches the answer very well.</td>
</tr>
<tr>
<td>3: Neutral</td>
<td>The question somewhat matches the answer.</td>
</tr>
<tr>
<td>1: Very Poor</td>
<td>The question does not match the answer.</td>
</tr>
</tbody>
</table>

Since it is time-consuming to evaluate the quality of questions manually, we only ask three English speakers to rate the scores on a limited number (30) of generated questions. Then, the speakers are asked to evaluate the Correctness and Relevance according to the rating scheme. To evaluate the Correctness, speakers are given questions generated from Du’s model and our ACNN model. In the evaluation of Relevance, both the answer sentences and the generated questions are provided.
5.7 Evaluation Results

5.7.1 Automatic Evaluation Results for ACNN

The experiment results between the ACNN model and Du’s model in terms of SQuAD, Electronics, Sports, and Home and Tools datasets are given in Table 5.9, Table 5.10, Table 5.11, and Table 5.12. The experimental results of the BLEU-1, BLEU-2, and ROUGE-L scores for all comparison methods are shown in the tables. The best outcomes between the baseline comparison as well as the ACNN variant proposed are highlighted in both.\(^6\)

**Table 5.9:** The comparison results in SQuAD.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du</td>
<td>43.09</td>
<td>25.96</td>
<td>39.75</td>
</tr>
<tr>
<td>ACNN</td>
<td>44.78</td>
<td>26.83</td>
<td>41.08</td>
</tr>
</tbody>
</table>

**Table 5.10:** The comparison results in Electronics.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du</td>
<td>12.99</td>
<td>7.08</td>
<td>24.19</td>
</tr>
<tr>
<td>ACNN</td>
<td>14.67</td>
<td>7.56</td>
<td>23.9</td>
</tr>
</tbody>
</table>

**Table 5.11:** The comparison results in Sports.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du</td>
<td>13.48</td>
<td>7.31</td>
<td>25.91</td>
</tr>
<tr>
<td>ACNN</td>
<td>14.88</td>
<td>7.94</td>
<td>25.95</td>
</tr>
</tbody>
</table>

\(^6\)Du's scores were cited from the article "Learn to ask: Neural Question Generation for Reading Comprehension" [1].
Table 5.12: The comparison results in Tools and Home.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du</td>
<td>13.95</td>
<td>7.73</td>
<td>26.75</td>
</tr>
<tr>
<td>ACNN</td>
<td>15.01</td>
<td>8.28</td>
<td>28.01</td>
</tr>
</tbody>
</table>

As Table 5.9 shows, ACNN model slightly outperforms Du in terms of the BLEU-1 score by 1.7 points. It also shows better scores in terms of BLEU-2 and ROUGE. These comparison results between ACNN and Du validate the effectiveness of adding the copy component.

The comparison results between our system and Du model when using the Amazon datasets are shown separately in Table 5.10, Table 5.11, and Table 5.12. We can see that our model performed slightly better than Du's model in all BLEU scores.

However, if we compare all the scores based on three metrics, the BLEU-1 and BLEU-2 scores with the SQuAD dataset are much higher than with the Amazon datasets. The main reason for this is the datasets themselves. SQuAD contains a corpus from Wikipedia articles that are generated by humans. Consequently, the sentences in this dataset have correct grammar and are generally well-structured. In contrast, the Amazon datasets originate from a real community question-answering pool. Therefore, there tends to be a lot of useless information in the sentences. As a result, these datasets contain significant “noise” that can affect the experimental results.

Another reason for the lower scores is that there are few overlaps between the questions generated by our system and ground truth questions. Table 5.13 features an instance from the Amazon-Electronics. In the answering sentence, the customer states a truth that the lens is compatible with a crop sensor. In the ground truth question, another customer asks whether the lens is compatible with a specific camera model number. Thus, the generated question asks whether the lens is compatible with another camera model number. Clearly, both questions are reasonable, grammatically correct, and readable. However, the score of this generated question is low because of the low overlap with the ground truth. This problem of low overlap
with the ground-truth questions is also mentioned in Yuan et al. [13].

To better evaluate our model’s results, we used several sample output sentences. Figure 5.3 compares sample output questions between our model and Du’s model. The copying word is highlighted in yellow, while “GT Questions“ indicates the ground-truth questions. We can see that our model’s generated questions show greater consistency with the answering sentence as well as the copying word. In addition, in Answer 6, our model generated a question with greater similarity to the ground-truth question than the question of Du’s model. This demonstrates our model’s effectiveness with the copying mechanism. In Answer 7, we can see that according to the answer sentence, Du’s model also generated a reasonable question, although it is not identical to the ground truth. This example is a good illustration of the reason behind the low overlap.

Table 5.13: Example of generated question and answer from Amazon datasets.

| Answer                                      | This lens is compatible with all Sony crop sensor DSLRs.  
|                                             | So yes it’ll work with your camera. It’s a great budget prime lens.  
| Ground Truth Question                       | I’m using the a37, so is this compatible with my camera?  
| Generated Question:                         | Is this lens compatible with the canon rebel t3i?  

### 5.7.2 Impact of Review Length

We also studied the impact of review length upon our ACNN model performance, using the SQuAD dataset. We tested different lengths using two values of 100 and 150. The comparison outcomes are shown in Table 5.14. It can be seen that as the length of the paragraph increases, the test score will drop. Through more contextual information is involved in longer paragraphs, they often contain more useless and irrelevant information.
Table 5.14: The results with different review lengths.

<table>
<thead>
<tr>
<th>Answer Length</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>44.37</td>
<td>26.15</td>
<td>40.57</td>
</tr>
<tr>
<td>150</td>
<td>43.97</td>
<td>25.63</td>
<td>39.95</td>
</tr>
</tbody>
</table>

5.7.3 Human Evaluations

To further analyze our model’s performance, we conducted evaluations of human upon the outcomes of our system as well as of Du’s model. Three English speakers were invited to mark the score of these sentences based on criteria including correctness and relevance. Correctness criterion is used to judge the generated questions’ grammar and fluency. Relevance evaluates whether the original text is relevant to the generated questions. The output score ranges from 1 (very poor) to 5 (excellent). Table 5.15 illustrates the human evaluation results of both our model as well as Du’s model according to the two primary criteria. Bold numbers denote superior performance based on the criteria.

Overall, the performance of the models by human scores has similar trends with automatic scores in aspects of BLEU and ROUGE. In addition, the correctness scores for both models are higher than the relevancy scores. It means that the question generation model tends to generate fluency and well-structured sentences comparing to identify relevant information in the context.

Table 5.15: Human evaluation results between our model and Du’s model.

<table>
<thead>
<tr>
<th>Models</th>
<th>Correctness</th>
<th>Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Du</td>
<td>4.3</td>
<td>2.93</td>
</tr>
<tr>
<td>ACNN</td>
<td>4.5</td>
<td>3.56</td>
</tr>
</tbody>
</table>

5.7.4 Summary Score Impact

Table 5.16 shows the results for ACNN and ACNN incorporating the summary score in the Laptop category. ACNN denotes the results for the ACNN model, while
ACNN+Repre indicates the results with incorporation of the summary score. We use boldface to highlight the best performance in each evaluation metric. We can see that the ACNN model that incorporates the summary score achieved better results than the model without it. However, the experiment on summary score impact is still preliminary and limited which can be further explored in the future.

### Table 5.16: Results of incorporating summary score in Laptop

<table>
<thead>
<tr>
<th>Models</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACNN</td>
<td>0.268</td>
<td>0.199</td>
<td>0.228</td>
</tr>
<tr>
<td>ACNN+Repre</td>
<td><strong>0.285</strong></td>
<td><strong>0.202</strong></td>
<td><strong>0.236</strong></td>
</tr>
</tbody>
</table>

To better evaluate the summary score’s impact, we offer a sample answer section from a laptop review\(^6\) of an Acer Aspire E 15. Table 5.17 and Table 5.18 show the ten questions generated by ACNN and ACNN incorporating the summary score. In this review, the customer first introduced the laptop’s overall product features, including battery and SSD, which are related to the first, second, fifth as well as sixth questions in Table 5.17 and the first, fifth and sixth questions in Table 5.18.

Then, the customer discussed his/her experience with connectors and watching movies, which are contained in question three and question four generated by both generators.

However, the major differences of the two models’ generated results are the sections illustrated within Figure 5.4, which refer to the display and the speakers. The last two questions generated by ACNN incorporating the summary score make more sense because they ask about the laptop’s display and customers’ experiences with the speakers in Table 5.18, while in Table 5.17 there are no questions related to these topics.

\(^6\)https://www.amazon.com/gp/customer-reviews/R35BV51W9KZ3ZZP/
Overall, after incorporating the summary score into the ACNN model, the final question-based summary contained highly detailed questions regarding battery life, SSD, connectors, and customers’ experiences with the speakers and when watching movies.

**Table 5.17:** Generated Questions by ACNN in Acer Review.

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the battery life?</td>
</tr>
<tr>
<td>Does this laptop have bluetooth?</td>
</tr>
<tr>
<td>Can I have DVD movies on it?</td>
</tr>
<tr>
<td>Does it have an HDMI port?</td>
</tr>
<tr>
<td>My intel i3/6gb/1tb hdd turns itself off during the nite every nite so</td>
</tr>
<tr>
<td>it’s not doing updates. all my settings are correct. anyone had this? why?</td>
</tr>
<tr>
<td>What kind of ssd does it take?</td>
</tr>
<tr>
<td>Will this laptop be able to run fortnite ?</td>
</tr>
</tbody>
</table>

**Table 5.18:** Generated Questions by ACNN+Repre in Acer Review.

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the battery life?</td>
</tr>
<tr>
<td>How is this laptop for viewing movies/videos?</td>
</tr>
<tr>
<td>Does it have an HDMI port?</td>
</tr>
<tr>
<td>Is it glossy display?</td>
</tr>
<tr>
<td>Can i add an ssd drive and still keep the hdd drive?</td>
</tr>
<tr>
<td>What kind of ssd does it take?</td>
</tr>
<tr>
<td>In the description there is no mention of external audio options.</td>
</tr>
<tr>
<td>I need a R&amp;L port for speakers. Does it have those?</td>
</tr>
</tbody>
</table>

### 5.8 Conclusion

With this chapter, our experiment’s dataset and the implementation details including the pre-processing steps were described. We also introduced and provide in-depth
discussions of our automatic and human evaluation metrics. We then presented the baselines for comparing our ACNN model taking different datasets into account. Finally, we analyzed the experimental results for our system and the selected baselines.
Answer 1: Powdered goat milk is still milk and therefore has lactose.
GT Question: Is this lactose free?
Du’s model: what are the ingredients?
Ours: is this product lactose free?

Answer 2: Nothing out of the ordinary, just a simple screw driver. If I recall correctly, I think it may have came with the tools needed to assemble. Good luck and congratulations.
GT Question: What tools are required to assemble unit?
Du’s model: What is the assembly required?
Ours: what tools do I need to assemble this?

Answer 3: You can definitely still do pushups with the wraps on. The wraps just give extra support, they really don’t impact your range of motion at all.
GT Question: Can I do pushups while wearing these wraps, or is the material too stiff?
Du’s model: Can you still use the material while wearing the material?
Ours: Can I do pushups while wearing these wraps?

Answer 4: I would go with a medium it fits well and when you adjust it with the helmet it’s tight to the chin.
GT Question: What size to buy for 14 yr old 125lb and 5‘5”?
Du’s model: I’m a woman with a small head. What size should I get?
Ours: What size should I get for a child who is 5‘6”?

Answer 5: There’s the ability to forward the bp measurement information via email to friends, family and doctors so I assume that once it’s been sent an email you can print it however I haven’t tested this functionality yet. At the very least when you bring up the bp readings on your screen you can do a screen capture and then print that screen capture.
GT Question: Is it possible to print the BP readings?
Du’s model: What is the difference between the BP and the BP?
Ours: How do you print from the BP?

Answer 6: I haven’t used the touch functions, but yes, I got video in windows 7.
GT Question: This monitor supports windows 7?
Du’s model: Does it have a built in Webcam?
Ours: Does it work with windows 7?

Answer 7: Lightweight. It can be lifted easily when folded up.
GT Question: What is the weight?
Du’s model: Is it hard to fold up?
Ours: How much does it weigh?

Figure 5.3: Sample output questions of our model and Du’s model. The copying word is highlighted in yellow.
The laptop display is good but has a poor viewing angle. I wasn’t surprised, my guess is that this is the display was where trade-offs were made to get the laptop to this price. The display is hard to see from the side (a witty salesperson would claim this is great for privacy) and required tilting the display forward/back more than I liked to obtain the best viewing angle from the front.

The speakers surprised me. The speaker volume is typical to most laptops in that I wish it would be about 20% louder at max. The speakers seems to be nice but not amazing, at least until I was lying on a couch watching a movie with the laptop on my chest. The nearness of my ears to the speakers allowed me to hear a very nice spatial mix of the sound. the movie sound track “bloomed” at that point and was surprisingly good. Bear in mind, I practically had to have the laptop a foot or two from my face for this to kick in.

**Figure 5.4:** Example of an answer section of one Acer Laptop Review from Amazon.
Chapter 6

Conclusion

6.1 Summary and Conclusion

Within this thesis, we seek to combine summarization and question generation problems. Questions are always highly attractive towards readers than long sentences. In addition, generating questions that are representative of long product specifications is a novel idea. Previous works have focused on extracting problems from existing question bases.

Throughout the research process, we proposed a two-stage framework consisting of QG and choosing the most representative questions. In the QG stage, we propose a novel QG model named Adaptive Copying Neural Network (ACNN). Our model employed a bidirectional Long Short-Term Memory (biLSTM) model with a copying mechanism that enhanced the connection between the original text and the generated questions. After generating questions, we calculated a summarization score by incorporating answer sections into the original text.

We first verified our ACNN framework using two different datasets: SQuAD and Amazon Question/Answer datasets. According to our experimental results, our original model outperforms previous models when evaluated by automatic evaluation metrics. In addition, our model’s generated questions obtained higher scores in human evaluation, indicating that our questions are more natural for people to understand. We then tested the questions with summarization scores by running the model again. Our preliminary statistical analysis showed that the questions generated by incorporating summarization scores performed better.
In conclusion, our model is fully data-driven and effective, thereby saving time and labor.

6.2 Future Work

This thesis represents an initial attempt to summarize generated questions in the area of natural language processing. Three potential directions should be pursued to further this work.

First, our primary task has been focused on single-document summarization through questions. This should be extended to multi-document summarization problems. Multiple documents could be treated as several answer sections that are identical to our model.

Second, it would be possible to change the structure of our encoder and decoder from a bidirectional LSTM to a reinforcement learning model, which has also shown superior performance in many natural language processing tasks.

Finally, we would like to apply our framework to many other tasks, such as the creation of educational materials and the generation of captions from images and videos.
List of References


