Baghaee, Tina

2017

Automatic Neural Question Generation using Community-based Question Answering Systems

Mathematics and Computer Science

AUTOMATIC NEURAL QUESTION GENERATION USING COMMUNITY-BASED QUESTION ANSWERING SYSTEMS

TINA BAGHAEE
Bachelor of Science, Shahid Beheshti University, 2011

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

AUTOMATIC NEURAL QUESTION GENERATION USING COMMUNITY-BASED QUESTION ANSWERING SYSTEMS

TINA BAGHAEE

Date of Defence: December 11, 2017

Dr. Yllias Chali Supervisor	Professor	Ph.D.
Dr. Wendy Osborn Committee Member	Associate Professor	Ph.D.
Dr. Jackie Rice Committee Member	Professor	Ph.D.
Dr. Howard Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

This work is dedicated to my supportive, wonderful family.

Abstract

In this thesis, we address the problem of opinion question generation. The motivation behind this task is to provide users with more question samples related to their query when using search engines. In our view, one of the datasets that is closest to peoples' thoughts, informal and casual speech are Community Question Answering (CQA) forums, where one can post questions, and other users can answer them. Specifically, we perform experiments on the Amazon question/answer dataset.

Unlike the conventional approaches that have tackled the question generation problem with hand-crafted rules, our approach is entirely data-driven. We model our problem with the sequence to sequence approach using an encoder-decoder structure, which has shown significant improvement in different natural language processing research areas in recent years. Our model benefits from the attention mechanism, which assists the model in focusing on a specific part of the input sentence. Furthermore, we provide solutions to the following problems: repetition of words and generating outside of vocabulary tokens. We provide a detailed explanation of the performance of the system. Experimental results show an improvement in automatic evaluation metrics such as the BLEU score over the state-of-the-art question generation system.

Acknowledgments

First and foremost, I would like to thank my supervisor, professor Yllias Chali who always encouraged me to work hard and provided an excellent environment for research with his support and guidance.

I would also like to thank my M.Sc. supervisory committee members Dr. Wendy Osborn and Dr. Jackie Rice for spending time reading my thesis and providing valuable feedback.

I thank my beloved family who encouraged me throughout my studies at the University of Lethbridge. Without their love and support, I would not have been here and could not finish this work.

I am thankful to my fiance, Amir, who always believed in me and challenges me to be better than myself. Finally, I am grateful to my best friends Robab, and Hossein who are like my older sister and brother and their love and guidance made my life as a student much more pleasant.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays when people need to acquire more information about a topic, one of the most convenient ways that comes to mind is to use the Internet. The amount of data on the Internet is so significant that it is becoming one of the most popular sources of information. Tools that make the process of finding the information much more straightforward are the search engines such as Google, Bing or Yahoo. People can obtain answer to their questions in a second. With these technologies, the difficulty of going to a library, searching through books and spending significant amounts of time scanning them for a specific answer no longer exists.

When someone enters a query in a search engine, thousands of links for that specific query will be provided. However, some of those links might be close to the query but do not provide the exact required information that provide an answer. The second stage of finding the answer to a query is for the user to go through all those links and find the desired answer. The process can quickly become frustrating if the user did not enter a correct query and obtains many unwanted results. In addition, going through each of these links and searching for facts can be highly time-consuming. Therefore entering a correct, informative query is required to obtain the desired results. Suppose the user wants to know more detail about the *Titanic* movie. If they only enter "titanic" as the query, the search engine may bring up the movie's trailer, cast members, and its filming location, or it might provide links regarding the real incident that happened in 1912, the number of casualties and

so forth. The user might spend significant amounts of time browsing through these links, skipping the most interesting parts and at the end become disappointed for not finding the desired results. According to Hasan (2013), due to their forgetful nature, people sometimes can not state precisely what is on their mind and therefore, the results that they obtain might differ from their intention. One solution to this problem is to provide some sample questions associated with the user's query. The user can either select one from those suggestions, or with the help of the suggestions, come up with a more exact query that is closer to their intentions. This is one of the primary motivations behind our question generation (QG) system; to suggest related sample questions to the users when they enter a query. In general, the goal of a question generation system is to produce a natural question given a sentence or a paragraph.

The majority of past studies on QG tackled this problem by preparing a set of hand-crafted, well-designed rules. They then transformed the input sentences into their syntactic representation and changed their declarative form to interrogative based on the mentioned rules. Our approach, on the other hand, does not involve any manually generated rules. Instead it is data-driven and requires at least 100k question/answer pairs as its input. For our approach, we need a dataset that is close to people's thoughts and is not in a strict written English form. When someone thinks of a query usually what they enter is informal, and may even contain colloquial expressions. In our opinion, the closest dataset to the mentioned conditions can be found in question and answer communities. *Yahoo! Answers*[1], *Quora*[2], *StackOverflow*[3] and *StackExchange*[4] are some of the popular communities that some people might browse through every once in a while. In these websites, people can post their questions about various topics, and other users can answer them based on their experience and knowledge. For our task, we decided to use the Amazon question/answer dataset (Wan and McAuley, 2016), which is based on Amazon products, and includes

[1] https://answers.yahoo.com
[2] https://www.quora.com
[3] https://stackoverflow.com
[4] https://stackexchange.com

21 different product categories. Therefore, our model contains a wide variety. We name our system **opinion question generation** since it works with a community-based question answering system as its input.

QG systems can also help to provide robust input for other tasks in the natural language processing research areas, such as question answering and reading comprehension task. In the question answering problem, the goal is to find concise answers in natural language to questions and the reading comprehension task aims to answer specific questions given a passage. By forming well-designed questions, a QG system can improve the inputs and provide a through dataset for such systems, therefore, spur the research in these tasks.

In general, QG can be considered to be a task which affects many aspects of a people's lives. Another considerable significance of QG is in its capability to improve one's learning ability. In a classroom students usually ask questions to compensate for their lack of information. On the other hand, teachers might pose questions during a lecture to make students aware of their information deficits, to give hints on the discussed subject, or sometimes just to assess the students' knowledge. Hence, a QG system can benefit teachers by facilitating their job and can be an aid to students during self-study.

Furthermore, medical chatbots which serve as intelligent health assistants and can ascertain someone's disease by asking questions about their symptoms can be improved by using a QG system. Other intelligent assistants such as Siri, which is an intelligent personal assistant for Apple users, or Cortana for Microsoft users can use the questions from an automatic QG system as inputs to their models.

We present a sequence-to-sequence model using an encoder and an attention-based decoder for the task of opinion question generation at a sentence level. The goal of the system is to aid search engine users by providing suggestions related to their query. For achieving the best performance, we enrich the system with effective features and propose practical solutions to the common problems in sequence learning models.

1.2 Contributions

Our work contributes to the question generation research area in the following ways:

- We introduce the opinion question generation task for assisting search engines by providing more example to the users.
- For implementing the opinion question generation task, We utilize a community question answering system as one of the closest sources to people's thoughts and questions. Specifically, we use Amazon question/answer dataset that consists of many categories.
- We benefit from the sequence-to-sequence approach by employing an encoder-decoder structure. In addition, attention mechanism is applied to further improve the model.
- We boost the performance of the system by incorporating coverage mechanism into our system. Many sequential models suffer from repetitive words in their output whereas our system prevents such repetition.
- We adopt input-feeding approach and address the problem of generating outside-of-vocabulary words.

1.3 Thesis Outline

The rest of this thesis is organized as follows.

Chapter 2: We demonstrate the intuition behind the approach we are adopting to address the question generation problem. This chapter also describes the background concepts necessary to understand the structure of our system.

Chapter 3: We provide a literature review on the previous proposed methods on question generation, followed by more recent studies that utilize sequence-to-sequence models. In addition, we briefly describe some of the research topics that employ an encoder-decoder framework in the natural language community.

Chapter 4: At the beginning of this chapter, we clarify the motivation behind our study as well as other work on this problem. We define the question generation task and describe our model as an encoder and attentional decoder. We highlight two issues that sequential models face: repetition of words and producing outside-of-vocabulary tokens. We propose effective solutions to resolve these problems. Furthermore, we explain the training and testing procedures.

Chapter 5: In this chapter, we provide an in-depth description of our dataset. We go through the detail in implementing the model. Additionally, we introduce the baseline system and describe all of the evaluation metrics that we use to compare our model to that system. Finally, we carry out a thorough explanation of the results.

Chapter 6: In the final chapter we conclude the thesis and suggest directions for future work in this research area.

Chapter 2

Background

2.1 Introduction

Deep leaning models are used in numerous research areas in computer science. Researchers have contributed deep learning-related work studies in several areas such as natural language processing (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Tu et al., 2016; Sordoni et al., 2015; Serban et al., 2016a; See et al., 2017; Du et al., 2017; Du and Cardie, 2017; Vaswani et al., 2017; Song et al., 2017; Zhou et al., 2017), computer vision (Nishani and io, 2017; Krizhevsky et al., 2012; Jia et al., 2014; Girshick et al., 2014; Szegedy et al., 2015), speech recognition (Graves et al., 2013; Graves and Jaitly, 2014; Abdel-Hamid et al., 2014; Nguyen et al., 2015; Sánchez-Gutiérrez et al., 2014), and bioinformatics (Angermueller et al., 2016; Lee et al., 2016; Zhen et al., 2017). The successful results of such studies motivated us to follow similar procedures for our task. We are applying deep learning techniques to address the problem of generating natural questions from users' reviews.

We first present a general understanding of the deep learning approach. We then go through the intuition behind deep neural networks and later expand this idea by describing more advanced models.

2.2 Deep Learning Outline

All the information regarding deep learning presented in this section is from (Goodfellow et al., 2016), which provides an introduction to several topics in deep learning tech-

niques used in both industry and academic research.

Over the past few years, with the advancement in different aspects of computer technologies, deep learning has made a vast improvement with providing better accuracy in complex applications. Availability of bigger size training data and enhancement in computer hardware and software like multi-core CPU/GPUs accelerated the growth of deep learning.

Deep learning is a technique in artificial intelligence which aids computers to learn from past experiences and conceive the world around them as a hierarchy of concepts. To elaborate, deep learning is a machine learning approach that simplifies working with data and provides flexibility and excellent power. However, unlike with basic machine learning techniques, there is no need for humans to describe all the required information for computers to perform a task. They can grasp all the necessary knowledge from past experiences.

For solving a lot of artificial intelligence problems, the usual routine is to extract some set of features and introduce them to a machine learning algorithm to optimize the weights on those features. Suppose we want to identify a person from characteristics of voices. This is referred to as a voice recognition task. One useful feature can be the size of person's vocal tract which clarifies whether the speaker is a man, woman or a child. On the other hand, if we want to state whether a car exists in a picture or not, we can consider a wheel existence as one of our features. However, due to other existing noises, finding a wheel as some pixel values might not be an easy task. Hence, feature extraction can be challenging.

Additionally, for numerous problems, sometimes there is a need for a sophisticated, human-level understanding of the raw data. Suppose we want to develop a speech recognition system. Accent, intonation, and pronunciation in speech are all factors of variation which can affect the raw data and need to be distinguished, which can make the process of extracting the high-level and abstract features difficult. The resulting features might be incomplete and over-specified, and designing and validating them is a time-consuming

process.

Deep learning deals with these problems by developing more complicated concepts from the simpler ones. If we visualize the creation of these concepts on top of each other, it can be depicted as a deep graph with several layers. That is the reason behind the word *deep* in this method's name.

The deep feedforward network or multi-layer perceptron (MLP) is a typical example of a deep learning model. MLP is a mathematical function that maps some input values to output values, and it consists of some other basic functions such as sigmoid and hyperbolic tangent function. MLP can be trained supervised (if we have labeled data) or unsupervised (if the input to the system is something like raw data).

2.3 Deep Neural Network

Many concepts and algorithms in deep learning were originally inspired by the functionality of the human brain and its learning process. Deep learning has been known by other names, such as artificial neural networks (ANNs) (Goodfellow et al., 2016).

The most common type of artificial neural network used in the majority of studies is a feed-forward neural network. The first layer in a feed-forward neural network is the input, and the last layer is the output. In between these layers are one or more hidden layers. Each layer consists of an arbitrary number of nodes. If the network has more than one hidden layer, it is called a *deep* neural network. According to Bengio (2009), representing some functions with a deep, hierarchical network can be more efficient compared to a shallow (one hidden layer) model. Delalleau and Bengio (2011) show this in practice by comparing a sum-product network in a deep network versus a shallow one. A basic feed-forward neural network is shown in Figure 2.1.

A node's responsibility in a neural network is to perform an operation, which involves multiplying some inputs with some coefficients or weights to increase or decrease the importance of that node. For instance, suppose we wish to assign labels to a set of images that

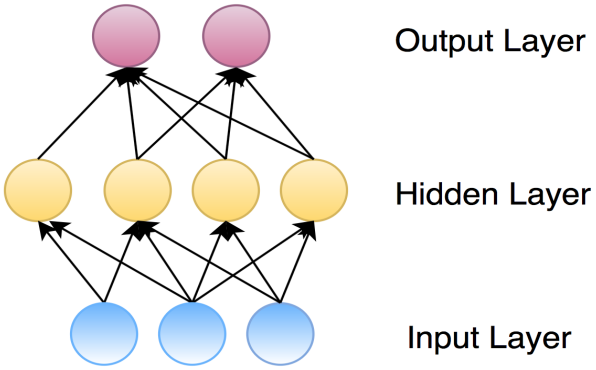


Figure 2.1: A basic feed-forward neural network (Socher, 2014).

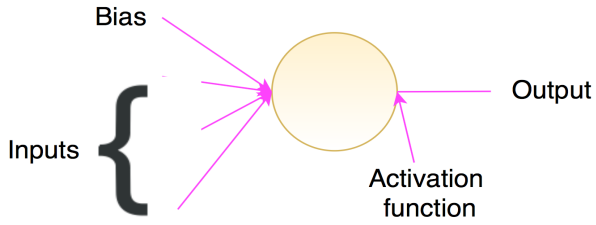


Figure 2.2: Structure of a single node (Socher, 2014).

state whether each is a picture of a car or not. The general goal in this example would be how to manipulate the weight of each input to decrease the error of classification.

Following explanations regarding the detailed structure of neural networks are from (Socher, 2014). Figure 2.2 illustrates the structure of a single node. It is a computational unit which consists of three inputs; a bias b, an activation function, and one output. If each input is an n-dimensional vector $x \in \mathbb{R}^n$, the output is calculated as follows:

$$a = f(w^T x + b). \tag{2.1}$$

Here, f is the activation function, defined as the hyperbolic tangent function in the example provided by (Socher, 2014):

$$f(x) = tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}.$$

The nodes in a neural network can be placed next to each other and on top of each other.

For a single node, Equation 2.1 becomes $a_i = f(W_i.x + b_i)$, where $W_i \in \mathbb{R}^n$ is the weight vector and shows how important the input x is and b_i is the bias. For simplicity, we can write this formulation for m nodes which are placed next to each other (stacked horizontally) in matrix notation (Socher, 2014):

$$z = Wx + b$$
$$a = f(z).$$

Here, $w \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and f operates element-wise:

$$f(z) = f([z_1, z_2, \ldots, z_m]) = [f(z_1), f(z_2), \ldots, f(z_m)].$$

If additional layers are added, the notation changes as in Equations 2.2. The superscripts denote the layer number (Socher, 2014).

$$\begin{aligned} z^{(2)} &= W^{(1)}x + b^{(1)} \\ a^{(2)} &= f(z^{(2)}) \\ z^{(3)} &= W^{(2)}a^{(2)} + b^{(2)} \\ a^{(3)} &= f(z^{(3)}) \\ &\cdots \\ z^{(l+1)} &= W^{(l)}a^{(l)} + b^{(l)} \\ a^{(l+1)} &= f(z^{(l+1)}). \end{aligned} \tag{2.2}$$

The last layer is then given to the output layer which in our case is a softmax function. The softmax function maps the input to a value between 0 to 1 such that the sum of all values will be equal to one.

2.4 Word Embedding

Suppose we have a vocabulary of five words: {python, java, ruby, car, vehicle}. In many traditional NLP tasks, a concept called *one-hot vector* is used to represent words (Olah, 2014). Values in one-hot vectors are zeros or ones, and the vector size is equal to the number of words in the vocabulary. Thus, in the above example each word vector has a dimension of five and a single 1 at its index location:

$$\begin{aligned} \text{python} &= [1\ 0\ 0\ 0\ 0] \\ \text{ruby} &= [0\ 0\ 1\ 0\ 0]. \end{aligned}$$

The problem with this representation is finding a similarity between words. If we calculate the dot product of these words, the results will be zero, which gives us no useful information and the model cannot comprehend the similarity between *ruby*, *java* and *python*. In addition, if the number of words in the vocabulary increases, the dimension of the one-hot vectors will be so high and working with a huge vocabulary will be difficult.

For calculating the similarities between words, we need further information. Instead of this simple method, Bengio et al. (2003) introduced an approach to learn word vector representation for predicting a word in a sentence.

According to (Olah, 2014), word embedding is a function which maps each word in a vocabulary to a high dimensional vector: $W : words \rightarrow R^n$. For instance,

$$W(home) = (-0.09, 0.16, 0.73, -0.37, -0.07, 0.76, ...).$$

W can be parameterized by a matrix. Specifically, each row of the matrix represents a word, and its corresponding vector with a particular dimension will be placed in the matching column. It can be thought of as a look-up table where each word is assigned to a vector within the table. In the beginning, W is initialized with random vectors for each word.

The intuition behind word embedding is measuring the similarities between words. Specifically, if two words are close in meaning, their word embedding will be similar to

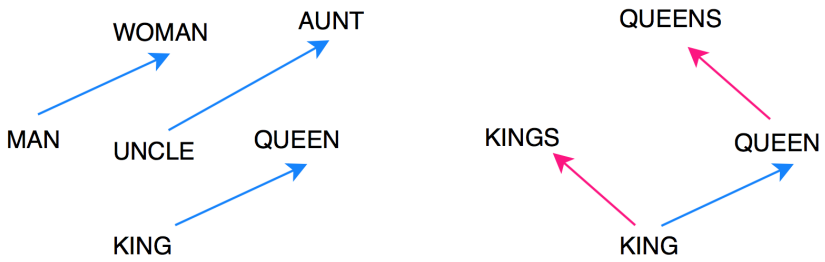


Figure 2.3: Visualization of vector offsets in gender relations as well as singular/plural relations (Mikolov et al., 2013c).

each other. For instance, consider the sentences "Lions live in the forest" and "Lions live in the jungle". The word embedding of *forest* and *jungle* should be close to each other. In addition, when there is a specific relationship between some words, there is a constant offset between their vector differences (Mikolov et al., 2013c). Consider the example below:

$$W(\text{“}man\text{”}) - W(\text{“}woman\text{”}) \simeq W(\text{“}king\text{”}) - W(\text{“}queen\text{”}) \simeq W(\text{“}nephew\text{”}) - W(\text{“}niece\text{”})$$

The male/female relation will remain the same in these pairs, and the word embedding consistently deals with the gender relationship (Olah, 2014). The same concept exists for syntactic relations as well. Singular/plural relations as well as gender relations are shown in figure 2.3.

In natural language processing, one can assign random vectors to the words in the vocabulary for constructing the word embedding and tune those weights during training or a use pre-trained word embedding. In the latter case, the word vectors are obtained from a model trained on a large corpus with billions of words. For instance, the Google pre-trained word2vec model[5] (Mikolov et al., 2013a,b) is trained on part of the Google news dataset that contains 100 billion words. The vectors are 300-dimensional and include 3 million words and phrases. Word2vec maps the words in the vocabulary to a vector representation where similar words have closer representation. Another example is Glove word embed-

[5]https://code.google.com/archive/p/word2vec/

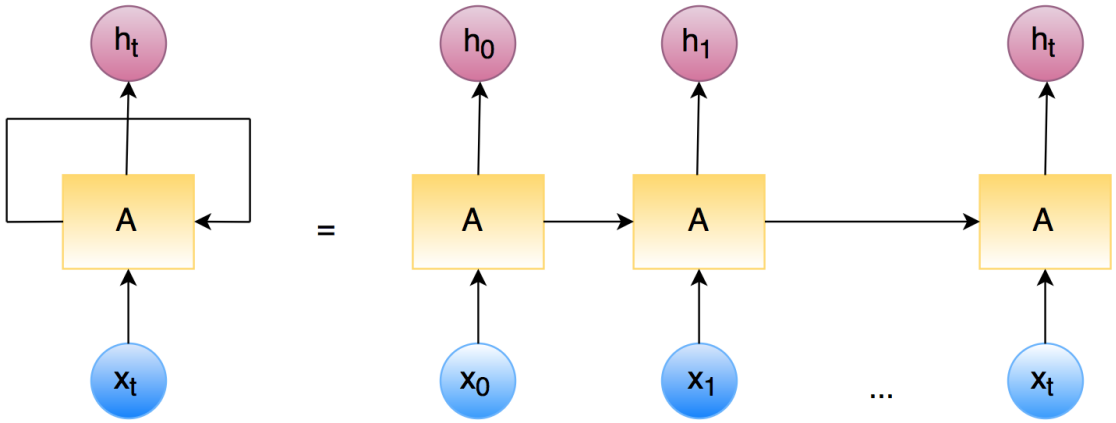


Figure 2.4: Recurrent neural network architecture (Olah, 2015).

dings (Pennington et al., 2014), which provide vectors trained on various corpora and with different dimensions.

2.5 Recurrent Neural Network

Consider the process of learning a complicated math problem. One does not all of a sudden understand the final answer. They have to go through the solution step by step. In order to learn the second step, they need the information that they have already understood from the previous steps. Traditional neural networks fail to simulate these kinds of sequential processes. They only consider the current input that is fed to them.

Recurrent neural networks or RNNs are used to deal with sequential data such as text sequences, speech, or videos, where the output is dependent on previous inputs (Olah, 2015). They have shown promising results in machine translation, document summarization, question answering, image caption generation and language modeling (Olah, 2015). RNNs have a loop inside their architecture which acts as a memory of what has been computed until this time step. The architecture of the recurrent neural network is shown in Figure 2.4.

At each time step, the input word x_t and previous hidden state h_{t-1} are fed to the RNN. x_t is the word embedding of the input at time step t. The current hidden state is calculated

as follows:

$$h_t = f(x_t, h_{t-1}), \tag{2.3}$$

where f is a nonlinear function. We elaborate on f in section 2.5.2.

2.5.1 Long-Term Dependencies Problem

We present the long-term dependencies problem with an example from Olah (2015). Consider the problem of language modeling. Given the sentence, "Lions live in the ... ", we want to predict the next word in the sentence. In this example, we do not need any further information. According to the word *lions* and *live* we can suggest words such as *jungle, forest* or *circus*. In this case, the distance between the required word and its relevant information in the sentence is not long. Therefore, learning this amount of information is not hard for the RNN.

On the contrary, consider this sentence: "Sarah is from Italy, and she grew up in there, she speaks fluent ... ". The final word should be a name of a language; however, we have to go back in the sentence to get more information in order to predict the language. Here, the gap between the required word and the relevant information *Italy* is long. The longer the gap, the harder it is for the RNN to connect the information.

2.5.2 LSTM Networks

To address the problem of long-term dependencies, Long Short-Term Memory or LSTM networks have been introduced by Hochreiter and Schmidhuber (1997). LSTM networks are a special type of RNN that can store the knowledge they have been exposed to for an extended period of time using gates that will be explained later in this subsection. LSTM differs from traditional RNNs by having memory cells c_t in their hidden layer. Figure 2.5 from (Graves, 2013) demonstrates the structure of a single LSTM memory cell.

We can employ a LSTM unit for the function f mentioned in section 2.5. The activation of LSTM unit at time step t can be calculated through the following equations:

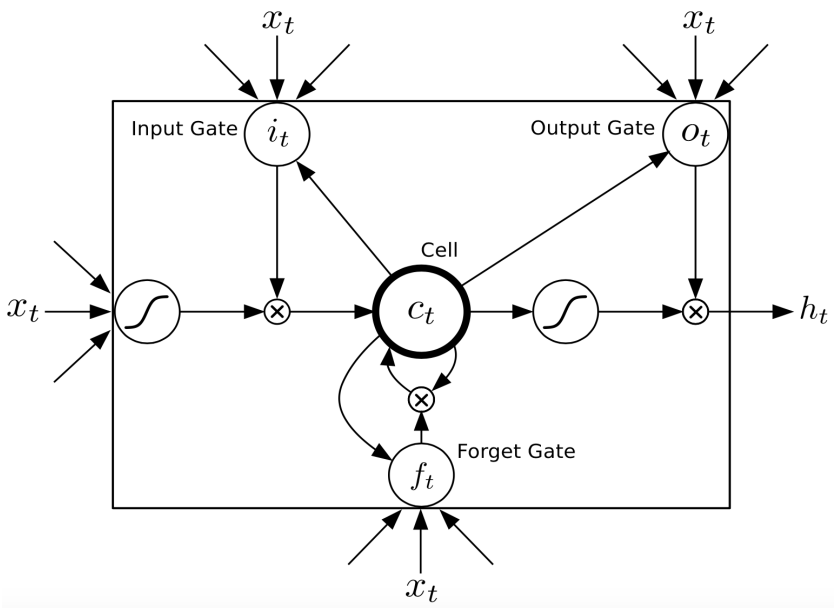


Figure 2.5: Long Short-Term Memory cell structure (Graves, 2013).

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.4}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.5}$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.6}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.7}$$

$$h_t = o_t \tanh(c_t). \tag{2.8}$$

For each unit, σ denotes the logistic sigmoid function and i_t, o_t, f_t and c_t represent an *input gate*, an *output gate*, a *forget gate* and a *cell memory* respectively. W is the weight matrix which is different according to the subscripts in each equation. For instance, W_{xf} indicates input-forget gate weight matrix and b is the bias term (Graves, 2013).

Intuitively, in Equation 2.7, the output gate filters the cell memory and calculates the output accordingly. The forget gate in Equation 2.5 is responsible for deciding what information should be kept and which ones should be disregarded. In Equation 2.6, the cell memory utilizes this information and updates itself by forgetting unnecessary data and

adding new memory content ($\tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$). Finally, the input gate in Equation 2.4 is responsible for deciding the amount of new memory content that will be added to the memory cell (Chung et al., 2014).

2.5.3 GRU Networks

A Gated Recurrent Unit (GRU) network is another type of RNN that has been introduced by Cho et al. (2014) to solve the long-term dependencies problem. These networks are motivated by the LSTM unit; however, their architecture is simpler. GRUs do not use a memory cell. They combine the input and forget gates into a single *update* gate z which is responsible for deciding how much of the previous memory should be kept. GRUs also use another gate called the *reset* gate r which decides how the new input and the past memory should be combined. The hidden state of the GRU at time t is calculated as follows:

$$
\begin{aligned}
h_t &= (1 - z_t)h_{t-1} + z_t s_t \\
z_t &= \sigma(W_z x_t + U_z h_{t-1}) \\
s_t &= \tanh(W_s x_t + U_s(r_t \odot h_{t-1})) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1}).
\end{aligned}
$$

In the above equations, $\odot$ is element-wise multiplication and W_z, U_z, W_r, U_r, W_s and U_s are weight matrices which are learned.

2.5.4 Bidirectional Recurrent Neural Network

Consider the problem of predicting a missing word in a sentence. To choose a viable option, an understanding of the words that are placed before and after the missing word is required. In other words, the output of the problem depends both on the former and future elements. For modeling such structures, Bidirectional Recurrent Neural Networks (Schuster et al., 1997) can be utilized. Bidirectional RNNs are two different RNNs that are

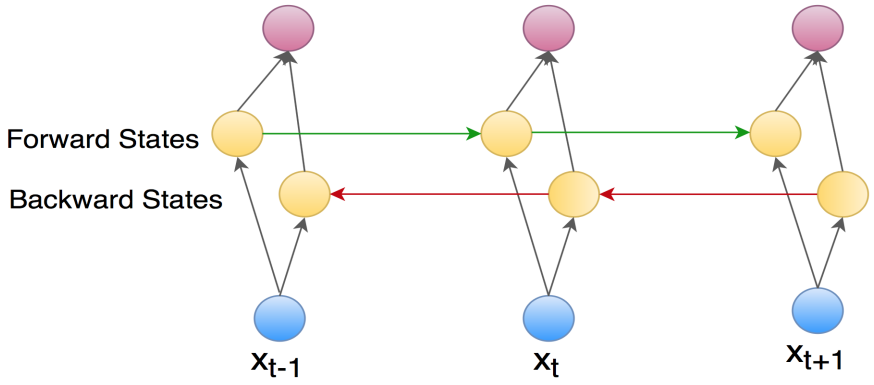


Figure 2.6: Unfolded structure of a bidirectional recurrent neural network (BRNN) for three time steps.

stacked on top of each other (Britz, 2015). One processes the sentence from left to right, and the other one performs the same task from right to left. The output depends on the hidden states of both RNNs. In our thesis work, we concatenate the outputs of the forward hidden states (from left to right) $\overrightarrow{b_j}$ and backward hidden states (from right to left) $\overleftarrow{b_j}$, namely, $b_j = [\overrightarrow{b_j}; \overleftarrow{b_j}]$ for input token j. Figure 2.6 illustrates the hidden states of the forward RNN and the backward RNN on top of each other for three time steps.

2.5.5 Stacked RNNs

To improve the power of a neural network model, we can increase the number of networks by stacking them on top of each other. In other words, we can stack multiple RNN, LSTM, GRU or any other neural network on top of each other. Consider the stacked RNN illustrated in Figure 2.7. Here, the outputs of each layer are fed to the next layer as its inputs. Specifically, the formulation at time step t is changed as follows:

$$h_{1,t} = RNN_1(x_t, h_{1,t-1})$$

$$h_{2,t} = RNN_2(h_{1,t}, h_{2,t-1})$$

$$\dots$$

$$h_{l,t} = RNN_l(h_{l-1,t}, h_{l,t-1}).$$

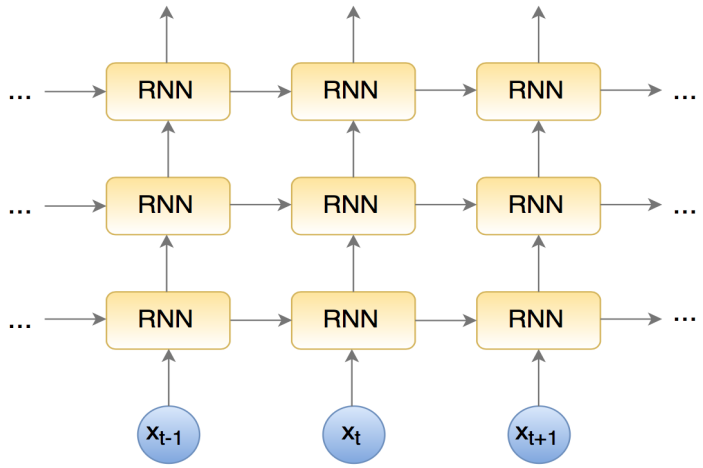


Figure 2.7: Structure of a stacked RNN

The RNNs can be replaced by any other recurrent model such as LTSM or GRU. $h_{l,t}$ denotes the hidden state of lth layer at time step t (Neubig, 2017).

2.6 Sequence to Sequence Learning

One of the limitations of deep neural networks is that their input and output must be a vector of fixed dimensionality. In many natural language processing problems, the length of the sentences is not known beforehand. Machine translation, speech recognition, question answering and generation are all examples of tasks with sequential data where a sequence of input words is mapped to a sequence of output words. Thus, there is a necessity of a standard technique to deal with these problems (Sutskever et al., 2014). For instance, in machine translation tasks, the goal is to map a sequence of words from the source language to its corresponding translation which is a sequence of words in the target language. The important point is that the source and target sentences are not the same length. To solve this issue, Sutskever et al. (2014) proposed a sequence to sequence learning model where two recurrent neural networks are used to process the input and output. The general structure of such a model is illustrated in Figure 2.8.

The model reads an input sentence "ABC", and it generates the output sentence "WXYZ". Clearly, the length of these two sentences is not the same. After the special end-of-sentence

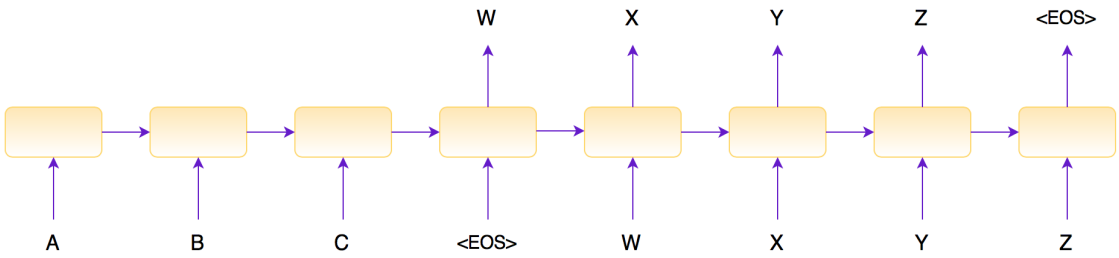


Figure 2.8: Sequence to sequence model where the length of the input and output sequences are different.

token "$<EOS>$" is produced, the model stops making predictions (Sutskever et al., 2014).

2.6.1 Encoder-Decoder Model

To implement the sequence to sequence learning approach, Cho et al. (2014) proposed the encoder-decoder model. This method also exploits two different recurrent neural networks to process the data, namely the encoder and the decoder (Cho et al., 2014). The encoder and decoder can be a RNN, a LSTM or a GRU. This model encodes the input sequence into a vector with fixed length and decodes a vector into a sequence of variable length.

The entire model is then trained to maximize the probability of a proper output according to the problem, given the input sentence. This output can be a reasonable question given a sentence in the question generation task, the correct translation in the neural machine translation task or the summary of a document given the entire input text in the text summarization problem. From a probabilistic perspective, the goal of the problem is to find the output sentence y such that the conditional probability of y given the input sentence x is maximized:

$$arg\max_{y} p(y|x). \tag{2.9}$$

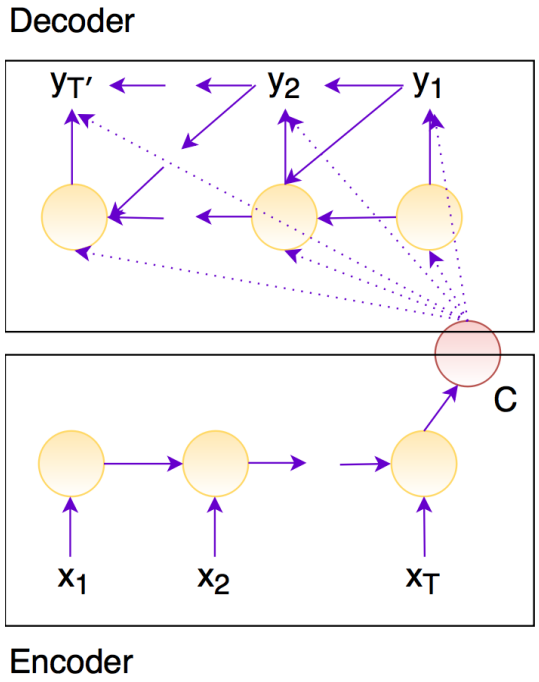


Figure 2.9: RNN encoder-decoder structure (Cho et al., 2014).

Encoder

The encoder is responsible for reading the input sentence sequentially via a RNN. Hence, the hidden states of the model change according to Equation (2.3) on page 13. f is a nonlinear activation function. It can be either a simple function such as logistic sigmoid function or a more advanced one such as a LSTM or a GRU. Encoding the input continues until the RNN reaches the "$<EOS>$" token. Then the vector c is generated from the hidden states by a non-linear function q (Bahdanau et al., 2015), which represents a summary of the source sentences. We elaborate on vector c in section 2.7. T_x denotes the length of the input sequence $X = (x_1, \ldots, x_{T_x})$,

$$c = q(\{h_1, \ldots, h_{T_x}\}) = h_{T_x}.$$

Decoder

The decoder is another RNN which is trained to produce outputs according to the previous hidden states and outputs, as well as the summary vector c. Thus, the hidden states of the decoder change according to the following equation at time step t:

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

where s_{t-1} is the previous hidden state. Figure 2.9 shows the general structure of the encoder-decoder model. The decoder decomposes the conditional probabilities in Equation 2.9 into ordered conditionals:

$$p(y|x) = \prod_{t=1}^{T} p(y_t|y_{t-1}, \ldots, y_1, \mathbf{x}).$$

Additionally, the conditional probability of the next output is computed as below where g is an activation function, and the outputs it produces must be normalized and in valid range. The Softmax function can be applied for this purpose (Bishop, 2006):

$$p(y_t|y_{t-1}, \ldots, y_1, \mathbf{x}) = g(s_t, y_{t-1}, c_t)$$

2.7 Attention Mechanism

One of the problems of the encoder-decoder structure is the need to define a fixed-length vector to store all the input sentence information. When the length of the input sentences increases handling these vectors will become harder. Specifically, Cho et al. (2014) showed that increasing the length of the input sentence affects the performance of their encoder-decoder model. Therefore, it can be difficult for the model to generate a reasonable output for longer input sentences (Bahdanau et al., 2015).

To address this problem, Bahdanau et al. (2015) introduced a model for neural machine translation task that does not limit the inputs to fixed-length vectors. Their model jointly

aligns and translate sentences. It encodes the input sentences into a set of vectors. During the decoding stage, it searches through these vectors to find the best positions that contain the most relevant information. A context vector is then generated according to these locations, and the model predicts the next word based on the context vectors and former generated words. As described in the section 2.6.1, the context vector c contains a summary of input sentences which generated by the encoder and is dependent on the hidden states of the encoder. The hidden state h_i of the encoder contains information about the entire input sentence with a focus on the words around the i^{th} entry. The context vector c_i is calculated as a weighted sum of encoder hidden states h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{2.10}$$

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

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik}))} \tag{2.11}$$

where e_{ij} is an alignment model, which measures how much the inputs at position j and the output around position i are matched. It is computed as:

$$e_{ij} = a(s_{i-1}, h_j). \tag{2.12}$$

The alignment model a is a feedforward neural network. It is trained jointly with all the other model parameters. In addition, e_{ij} establishes a mechanism of attention in the decoder (i.e., it assists the decoder in deciding which parts of input sentence to pay attention to.) Therefore, there is no need for the encoder to compress all of the required information into a fixed-length vector. Figure 2.10 illustrates the attention model from (Bahdanau et al., 2015).

Luong et al. (2015) extended this solution and introduced *local* and *global* attention. In global attention, the decoder pays attention to all source words, while in local attention, it

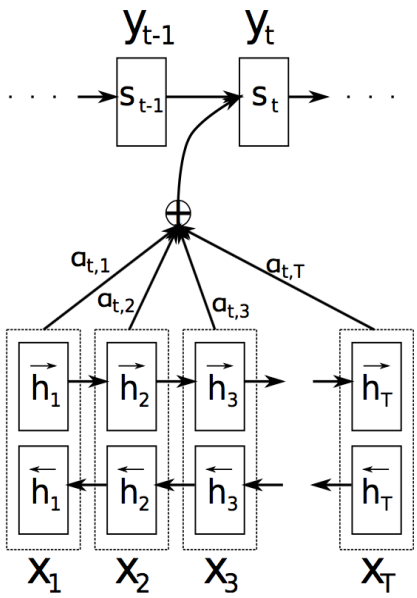


Figure 2.10: Bahdanau et al. (2015) proposed attentional model with bidirectional encoder.

only attends to a subset of source words at a time.

2.8 Training and Inference

After defining the details of the model, it must be trained on a large dataset. To train the encoder-decoder model, we need an objective. Consider a training corpus of input and output pairs: $S = \{(x_i, y_i)\}_1^{|S|}$ where $X = (x_1, x_2, \ldots, x_N)$ is the input and $Y = (y_1, y_2, \ldots, y_M)$ is the output. The training goal is to jointly train the encoder and decoder to minimize the negative log-likelihood of the training data concerning all the model parameters denoted by θ:

$$J_t = -\sum_{i=1}^{|S|} \log p(y_i|x_i;\theta).$$

After training the parameters of the model, the decoder can generate predictions for new inputs. It performs inference by utilizing a beam search to maximize the conditional probability in Equation 2.9 on page 19. In the following sections, we first describe the greedy search which is the most basic approach to the inference problem before describing

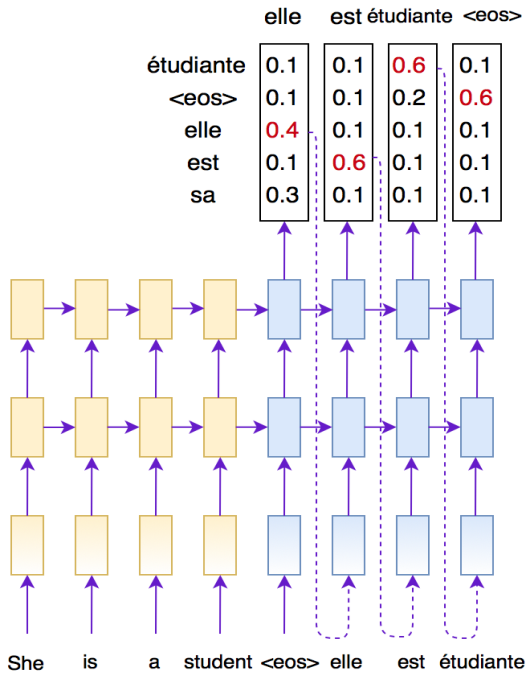


Figure 2.11: An example of inference with greedy decoding for English-to-French machine translation (Luong, 2016).

the beam search as the standard technique.

2.8.1 Greedy Search

As discussed before, the objective of the inference is to find the target sentence Y that maximizes the conditional probability $p(Y|X)$. The primary solution to find each target word y_t is to apply a greedy search, where $p(y_i|x_i)$ is calculated, and the word with the highest probability is selected as the next target word. Consider Figure 2.11 for the English-to-French machine translation task (Luong, 2016). In the first decoding step, *elle* has the highest translation probability, and it is selected as the input for the next time step. The process continues until the end-of-sentence token ("$<eos>$" in this example) is produced. The problem with greedy decoding is that it only finds the locally optimal solution and only generates one sample. therefore, we can not choose from several samples.

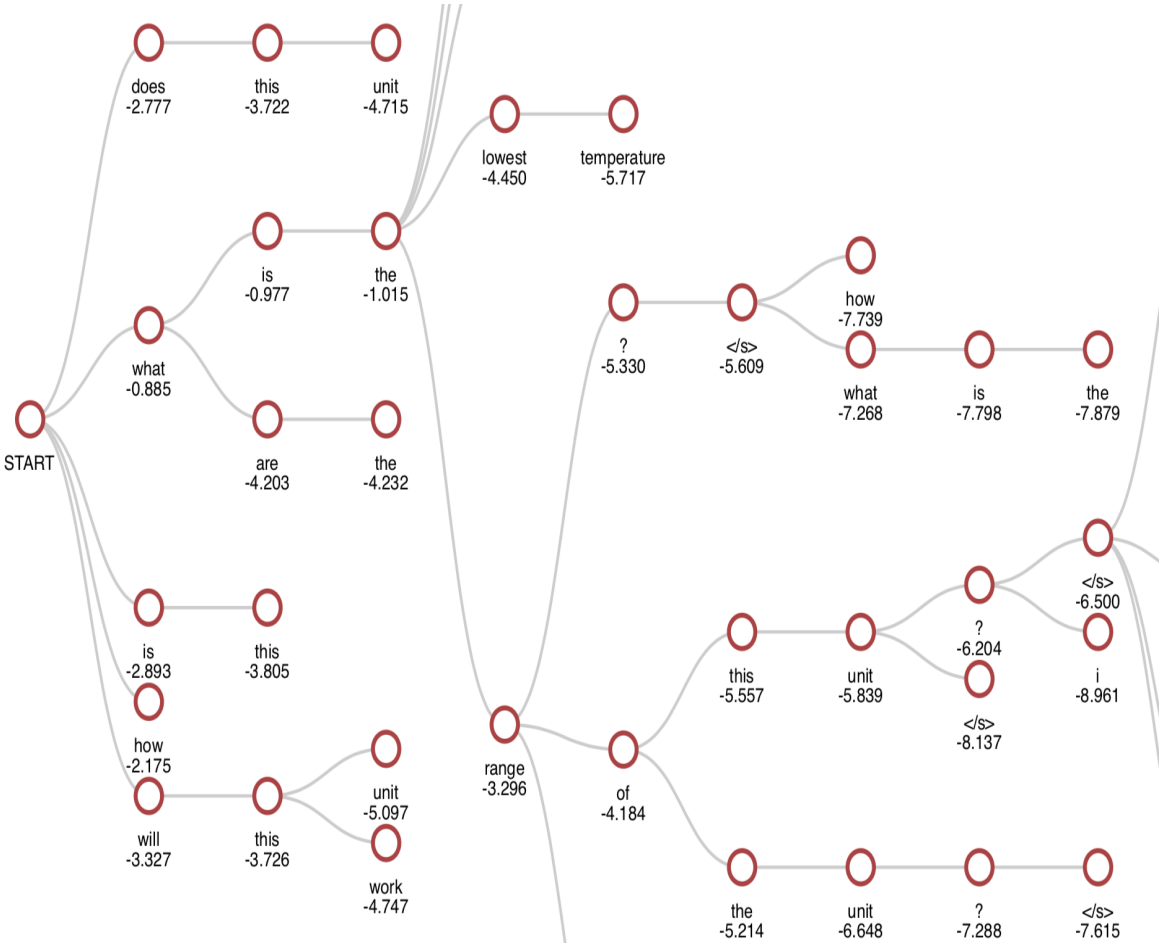


Figure 2.12: Beam search representation with size 5.

2.8.2 Beam Search

To address the problem of local optimum, many researchers have applied beam search (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). Unlike greedy decoding, which only keeps one best hypothesis during the decoding, beam search keeps $B > 1$ hypotheses or *beams* and picks the best one based on a scoring function. We call this B the beam size. The beam search utilizes breadth-first search for building its search tree. In addition, it generates multiple samples which we can choose from. The beam search continues until it produces "*EOS*" token or reaches the maximum length of the sequence. Figure 2.12 visualizes the beam search representation partially with beam size 5 along with the score of each node. The token "*</s>*" represents the end-of-sentence in this example.

2.9 Summary

In this chapter we reviewed the background required for understanding our question generation system. We first represented the general intuition behind the deep learning research area. We then described the main ideas regarding deep neural networks and introduced different types of neural networks such as RNN and LSTM, as well as different structures such as Bidirectional and stacked RNNs. We explained the word embedding and elaborated on sequence-to-sequence learning with an encoder-decoder structure. We demonstrated the attention mechanism and presented the training and inference procedures. In chapter 4 we will explain our proposed system in detail.

Chapter 3

Literature Review

3.1 Introduction

In general, methods for addressing the question generation (QG) problem fall into four different categories: syntax-based (Heilman and Smith, 2009; Wyse and Piwek, 2009), semantic-based (Sag and Flickinger, 2008), template-based (Chen et al., 2009) and end-to-end sequence-to-sequence learning models (Du et al., 2017; Yuan et al., 2017). Syntax-based approaches are more promising for short sentences but do not always guarantee grammatical correctness (Yao and Zhang 2010). Semantic-based methods employ target identification techniques such as semantic role labeling (Lindberg et al. 2013), while template-based approaches are more efficient in specific-purpose domains where human resources are usually needed to create robust templates (Yao and Zhang 2010). However, sequence-to-sequence learning approaches for solving the question generation problem do not rely on hand-crafted rules. These methods are not dependent on well-designed human-generated rules, which transform declarative sentences to interrogative ones (Du et al., 2017).

In the following section, different techniques for solving the question generation problem will be discussed. Motivations behind this task include facilitating the learning process for students, a boost in the performance of question answering systems, and aid for search engines. These are explained later in the chapter.

3.2 Traditional Approaches in the Question Generation Problem

Over the past few years, the question generation task has received tremendous attention from the natural language processing community, especially after The First Question Generation Shared Task Evaluation Challenge (Rus et al. 2010). This challenge consists of two parts: the first part is question generation from paragraphs and the second part, which received more interest, is question generation from single sentences.

As a follow up to this task, many researchers developed different techniques to address these problems. For the second task, Ali et al. (2010) developed a system that first converts the complex input text into a few simple sentences, and classifies each sentence for the proper type of question according to subject, verb, object, and preposition of the sentences. They did not utilize semantic information in their work.

There are several methods that address the question generation problem. Mitkov and Ha (2003) succeed in producing multiple-choice questions from electronic educational documents in addition to generating semantically related distractors. They made use of a corpus and WordNet as well as a shallow parser, automatic term extraction, and word sense disambiguation. For simplicity, they generate questions from declarative sentences by applying transformational rules to them.

In another study, researchers studied the vocabulary assessment problem. These tests are mostly time-consuming processes that consist of hand-written development and can be dealt with subjectively (Brown et al. 2005). Brown et al. (2005) developed a system that automatically generates vocabulary assessment questions. These questions are of 6 types and can have various forms, such as wordbank or multiple-choice. They also validate the correctness of generated questions.

Wang et al. (2007) utilized templates that had been trained on medical articles to automatically generate questions for the assessment of a person's understanding of reading material. The generated questions are factual and might not be so informative since they only use one sentence for the question generation.

Chen et al. (2009) expanded their previous work on automatic question generation from a narrative text (Mostow and Chen 2009), which then can be used for creating modeling and applying instructions. In their later work (Chen et al. 2009), the primary focus is on generalizing the former technique on informational texts to assist children in grades 1-3 with a better comprehension of informational texts.

One of the syntax-based methods for the QG problem was the work carried out by Kalady et al. (2010), which utilized keyword modeling. They produced factoid and definitional questions by applying named entity recognition, parse tree and identification of main phrases in a document (U-Keys). Consequently, they generated both yes/no and wh-questions.

Liu et al. (2010) developed a system to automatically generate trigger questions in support for students' writing literature reviews. In their procedure, first, the citations and core content elements are extracted from the students' writing. Later, by employing rule-based techniques, they classify the citations and generate questions according to a set of templates.

Piwek and Stoyanchev (2010) worked on the CODA project, which concentrates on the automation of dialogue generation from a monologue. Their focus was mainly on question generation as a part of dialogue generation for a better representation of educational documents.

Another study that took advantage of a template-based approach was done by Heilman and Smith (2010b). Their technique is to overgenerate questions and then rank them. To transform declarative sentences into questions, they produced some hand-written rules to perform a syntactic transformation of those sentences. Furthermore, they used a logistic regression model for the purpose of ranking. In another work, Heilman and Smith (2010a) addressed the QG problem on factual information in syntactically complex reading materials. In this strategy, they proposed an algorithm to extract simple, accurate factual statements from the semantic and syntactic point of view. They demonstrated that their system is more practical for factual question generation than a standard text compression algorithm.

Other some researchers worked in the area of cloze question (gap-fill question) generation. Their task includes generating gap-fill questions and providing one correct answer and three different distractors. Agarwal and Mannem (2011) achieved this task with the help of syntactical and lexical features. By using summarization techniques, they first find the informative sentences from the content and then choose the appropriate key to ask about. Finally, they pick a distractor according to its similarity to the key. Kumar et al. (2015) addressed the same problem. For the sentence selection stage, with the help of topic distributions gained from a topic model, they proposed a sentence ranking technique. For the gap selection phase, they utilized human annotations to train a classifier that predicts the relevance of gaps. Finally, to find relevant distractors, they applied the word2vec language model probabilities and the Dice Coefficient.

Agarwal et al. (2011) approached the QG problem by using discourse cues. Their technique consists of two parts: content selection and question formation. In the first part, they look for a proper sentence in the text to ask a question about, which in the second part they examine the sense of discourse connectives to select the suitable question type. In the end, they employ syntactic transformation to generate the final question.

Zheng et al. (2011) proposed a system that converts keywords into questions (K2Q). It takes both query history and user feedback into consideration. K2Q generates a set of candidate questions as well as a list of refinement words. The user either choose a preferable question or a refinement word. Then, the system generates another set of candidate questions along with refinement words until the user finds the desirable question or quits.

Becker et al. (2012) focus was on generating quiz questions that use online documents for self-motivated learners to learn about new areas. Their major challenge was finding a suitable part of a sentence to ask about, which is called a gap selection. Their study shows that all semantic roles can be good candidates for these gaps. They successfully produced a cloze (fill-in-the-blank) question generator.

One of the applications of question generation is engaging readers when reading a news

article. By presenting different questions in an article, a reader tries to answer those questions and therefore, focuses more on that topic. Rokhlenko and Szpektor (2013) developed a system to generate comparable questions in news articles. Their algorithm consists of an offline and an online part. In the offline step, a database of similar question templates will be created, while in the online part, the algorithm will pick a relative template according to the article content. Finally, the algorithm fills the template with the article entities in a way that the comparison between the entities seems logical.

Lindberg et al. (2013) employed a template-based approach while taking advantage of semantic information to generate natural language questions for online learning support. Mazidi and Nielsen (2014) also applied a semantic approach to create questions of different types and depths that can be used for self-studying or tutoring. Specifically, they used semantic role labels to create both questions and answers of a given text. Finally, they depicted three linguistic challenges: negation detection, coreference resolution, and verb forms. Another semantic-based approach was proposed by Yao and Zhang (2010). The primary idea behind their system is a Minimal Recursion Semantics (MRS) transfer. The first part of their strategy is MRS decomposition of complex sentences into smaller parts. Next, they map the semantic representation from declarative sentences into interrogative ones and generate questions according to a linguistically deep English grammar.

Labutov et al. (2015) developed a system which generates questions without having a detailed semantic information of the text. The primary idea behind their work is benefiting from a low-dimensional ontology for document segments. They then crowdsource a set of promising question templates that are matched with that representation. These templates can be reused in many documents. Finally, they rank the results based on their relevance to the original input.

Chali and Hasan (2015) generated questions from a topic of interest by utilizing named entity information and the predicate argument structures of a body of texts associated with that topic. They applied Latent Dirichlet Allocation to calculate the importance of the

generated questions. To achieve this, they identified the subtopics in the texts and measured their similarity with the questions by employing an Extended String Subsequence Kernel. To evaluate the syntactic correctness of the generated questions automatically, they applied syntactic tree kernels. Finally, they ranked the questions by considering both their syntactic correctness and their importance.

Chali and Golestanirad (2016) proposed a system that automatically generates comprehensive questions from paragraphs considering that each body of text is related to a topic of interest. They designed 265 templates and 350 rules to generate questions. They assessed the generated questions from the grammatical point of view using tree kernel functions. To measure the importance of the generated questions, they ranked the questions using community-based question answering systems.

3.3 End-to-End Sequence-to-Sequence Learning Models

With the successful results of deep learning approaches in several natural language processing tasks, many recent studies have been performed to take advantage of these techniques. It has shown promising results in machine translation (Cho et al., 2014; Tu et al., 2016), text summarization (See et al., 2017; Rush et al., 2015), question answering (Song et al., 2017), and reading comprehension (Nguyen et al., 2016).

A recent work on the question generation task, which uses Neural Network concepts was done by Serban et al. (2016a). They address this problem, by converting knowledge graph facts into questions. As a result, they created a factoid question-answer corpus by using a Recurrent Neural Network architecture and demonstrated that their system outperforms the proposed template-based baseline.

There has been considerable work on the integration of the natural language and computer vision communities over the past few years. Image caption generation, video transcription and answering questions regarding an image can be named as some of the examples. Additionally, Mostafazadeh et al. (2016) introduced the visual question generation

task where the goal of the system is to create a question given an image.

Question generation can also be combined with its complementary task, Question Answering (QA) for further improvement. Tang et al. (2017) consider QG and QA as dual tasks and train their relative models simultaneously. Their training framework takes advantage of the probabilistic correlation between the two tasks.

Song et al. (2017) also proposed a model for both the QG and the QA problems. They applied an attention-based encoder-decoder model, which is dependent on the target answer, and takes both the passage and the target answer as its inputs. It then performs query understanding to capture more relations between the target answer and the passage. They also utilized a policy gradient learning algorithm to address the bias exposure problem during training and used copy and coverage mechanisms on the decoder.

Zhou et al. (2017) also used an attentional encoder-decoder framework which takes the answer position in the input sentence and lexical features as well as the passage as the inputs to the encoder. The lexical features that they applied are the part-of-speech (POS) and named entity (NER) tags.

One of the latest studies on question generation comes from by Du et al. (2017). They study the reading comprehension problem, which consists of question generation from both sentences and paragraphs. They adopt an attention-based sequence learning model. They later developed another model by improving the input data. Specifically, in (Du and Cardie, 2017) they present an approach to identify question-worthy sentences given a paragraph, and use those sentences as input for their former question generation model. For this task, they employed a hierarchical neural sentence labeling model and applied both sum and convolution operations for the encoding.

Another recent work is by Yuan et al. (2017). They generate questions from documents using supervised and reinforcement learning. Specifically, they apply policy gradient techniques to fine-tune their model and maximize the rewards used for measuring the quality of generated questions. The motivation behind their question generation system is improving

the performance of question answering systems.

3.4 Other Problems Motivated by Encoder-Decoder Model

There are other tasks in natural language processing for which their solutions can be a motivation for addressing the question generation problem. One of them is the work carried out by Sordoni et al. (2015). The problem they address is query suggestion, which is a generation task that produces a contextual suggestion close to the user query when searching online. With a neural network perspective, we can map the QG problem to this task; that is, generating a specific text (question) according to the input sentence (answer).

Sordoni et al. (2015) trained a hierarchical neural network model, which takes two similar queries as its inputs and samples a new query. Their method considers previous queries and their order while trying to avoid data sparsity. It also takes rare queries into account. In addition, they produce the proper query suggestions one word at a time and show that their technique outperforms other methods in this task.

Following this work, Serban et al. (2016b) applied the same generative hierarchical neural network model to the challenging task of dialogue response generation. Their work focuses on the non-goal-driven dialogue systems like language learning tools or computer game characters. Their method can be applied to a variety of sequential generation tasks, such as the QG problem. To improve the learning procedure, they pretrained the model on a larger question-answer pair corpus. The final results demonstrate a better performance on the dialogue generation task as compared to n-gram based and baseline neural network models. Serban et al. (2016c) extend the previous work by augmenting latent stochastic variables to their model. A Latent variable hierarchical recurrent encoder-decoder (VHRED) model showed the ability to generate long, diverse utterances with better quality compared to former models. The authors trained this model by maximizing a variational lower-bound on the log-likelihood.

In addition to the above mentioned works, there are several studies in the areas of neural

machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), text summarization (Rus et al., 2010; Iyer et al., 2016), image caption generation (Xu et al., 2015; Karpathy and Fei-Fei, 2017) and question answering (Rajpurkar et al., 2016; Nguyen et al., 2016; Yin et al., 2016) that apply sequence learning models.

3.5 Summary

In this chapter we reviewed the studies on traditional approaches that have been applied to the QG problem such as syntactic-based, semantic-based, and template-based methods. In addition, we presented works that used sequence-to-sequence learning models and finally, we described some other tasks in natural language processing that utilize the same techniques. In the next chapter we explain our proposed system in detail.

Chapter 4

Question Generation Problem

4.1 Motivation

One of the many unconscious activities that people engage in their everyday life is trying to deduce and make decisions by asking questions and answering them in their minds. Sometimes, they have sufficient knowledge to answer their needs, but it frequently happens that their lack of information makes them seek external help. There are many resources that people can acquire information from. One of the sources that play a significant role in today's world is the search engines. However, previous studies have shown that people are forgetful in nature and they may not be able to express their intention exactly as it crosses their minds (Hasan, 2013). Consequently, they may not receive the desired results based on the queries they provide. This is one of the motivations behind a question generation system. The goal of a Question Generation (QG) system is to create natural questions given a sentence. Such system can suggest a couple of question samples based on the query that the user enters. Generally what people enter as their query is their informal speech that may not follow the correct grammatical structure. This was the motivation behind the development of a question generation system for such data. For this purpose, we decided to utilize community-based question answering (CQA) systems. In our opinion, these systems are the closest with respect to the structure to what people enter as their query. To elaborate, a community-based question answering system is a platform where people post their questions, and other users can answer them. These services are becoming a useful knowledge source that are rich in insights, expertise and interactions (Blooma and

Kurian, 2011). We utilize the answers that people post on the CQA system as an input to our model; hence, proposing an **opinion question generation system**. The generated questions can be used later as suggested samples to users when entering their queries.

There are additional motives for developing a question generation system. One of the most primary reasons is the significance of such systems in the learning process. According to (Graesser and Person, 1994), question generation plays a prominent role in cognitive processes which operate at deep conceptual levels; namely, learning of complicated material, problem-solving, text and social action comprehension, and creativity. By asking good questions, students can improve their learning, perception, and memory of the complex material. It also helps students realize their knowledge deficits and encourage them to look for more information to compensate for those deficits. Boosting learning ability has been a motivation behind a lot of question generation systems (Chen et al., 2009; Liu et al., 2010; Becker et al., 2012; Lindberg et al., 2013; Mazidi and Nielsen, 2014).

One of the other benefits of a question generation system is its ability to make advancement in other research areas of natural language processing community, such as question answering and reading comprehension. In the reading comprehension task, given a passage, the goal is to answer some questions related to the information of that passage. A question generation system can provide a robust input for this task. The same concept exists in question answering task as well. This has been a motivation behind research carried out by (Serban et al., 2016a; Yang et al., 2017; Rajpurkar et al., 2016; Nguyen et al., 2016).

QG has also been used as a strategy for engaging users when reading news articles (Rokhlenko and Szpektor, 2013). Presenting an irresistible question, which can stir up a person's emotion would definitely make them keep reading the article. In our work, we propose a sequence to sequence model that use a coverage mechanism as a boost to an attention mechanism for addressing the question generation problem in sentence level, which we explain in next sections. In this chapter, we are going to present all the necessary details respecting our question generation system.

4.2 Task Definition

In this section, we define the question generation task. Our work is inspired by neural machine translation, which is one of the most important topics in the field of natural language processing (Neubig, 2017). Generally, our task is translating an answer sentence into its related questions.

Given an answer sentence $A = (a_1, a_2, ..., a_N)$ where a_i is a token and N is the maximum length of the answer, we are going to generate a natural question $Q = (q_1, q_2, ..., q_M)$ with the length M, which has its answer embedded in A. Our goal is to find Q such that the conditional probability $p(Q|A)$ is maximixed. Specifically, we are going to model $p(Q|A)$ as a product of word predictions:

$$p(Q|A) = \prod_{1}^{M} p(q_t|q_{1:t-1}, A)$$

$$p(Q|A) = \prod_{1}^{M} p(q_t|q_{1:t-1}, A)$$

This indicates that the probability of each q_t relies on the previously generated words and the input sentence A.

4.3 Model Structure

For modeling $p(Q|A)$, we use the encoder-decoder architecture (Cho et al., 2014) where there are two recurrent neural networks: the encoder, which reads the input sequence one word at a time and converts it into a vector representation, and the decoder, which generates the output sequence according to the encoder's output and all previously generated words. In the following sections, we are going to describe our encoder and decoder architectures precisely.

4.3.1 Encoder

An encoder network is a RNN that maps an input sequence into a word vector and then converts it into hidden states $h_1, ..., h_N$. The hidden states of the encoder are computed as:

$$h_t = \text{LSTM}(e(a_t), h_{t-1}),$$

where $e(a_t) \in \mathbb{R}^m$ represents the m-dimensional word embedding of the word a_t and h_{t-1} is the previous hidden state.

Unidirectional encoders read the input from left to right and only summarize the information of the previous words, they do not have any knowledge regarding the future words. To make the model aware of the future words, we employ bidirectional LSTM as suggested in (Cho et al., 2014), which consist of the forward and backward LSTMs placed on top of each other. The hidden states of the forward pass and the backward pass LSTM at time step t are computed as follows:

$$\overrightarrow{h}_t = \overrightarrow{\text{LSTM}}(e(a_t), \overrightarrow{h}_{t-1})$$
$$\overleftarrow{h}_t = \overleftarrow{\text{LSTM}}(e(a_t), \overleftarrow{h}_{t+1}).$$

The initial hidden state of the encoder is set to zero; thus, $\overrightarrow{h}_1 = 0$ and $\overleftarrow{h}_N = 0$. We concatenate the outputs of the forward hidden states $\overrightarrow{h_t}$ and the backward hidden states $\overleftarrow{h_t}$ to obtain the annotation h_t, which shows the final hidden state of the encoder :

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t}, \tag{4.1}$$

where $\oplus$ denotes the concatenation operator.

To make our model more powerful we are going to stack two LSTM layers on top of each other. The hidden states of each LSTM layer is computed as:

$$h_{1,t} = \text{LSTM}_1(e(a_t), h_{1,t-1})$$
$$h_{2,t} = \text{LSTM}_2(h_{1,t}, h_{2,t-1}).$$

Again the initial hidden states are: $\overrightarrow{h}_{1,1} = 0$ for the forward pass and $\overleftarrow{h}_{1,N} = 0$ for the backward pass. Furthermore, we concatenate the hidden states of the forward and backward pass of the encoder to initialize the decoder's hidden states:

$$s = \overrightarrow{h}_{2,N} \oplus \overleftarrow{h}_{2,1}.$$

4.3.2 Attention-Based Decoder

We are using an attention-based decoder from (Luong et al., 2015) which allows the decoder to learn to focus on a particular range of the input sequence during the generation task. This is similar to the process of forming a question in a human's mind, where one pays attention to a specific part of a sentence and creates a question regarding that part.

The decoder itself is another LSTM that takes the encoder output and creates a sequence of words as the question. In the simplest form of the decoder, only the last output of the encoder, which is called the context vector, is used to initialize the hidden state of the decoder. In this way, the context vector is responsible for encoding the entire sentence. To ease the burden for the context vector, we apply an attention mechanism. Attention helps the decoder to focus on different parts of the encoder's output. The output of equation (4.1) is used to calculate the context vector c_t. Here, for preventing confusion between the hidden states of the encoder and the decoder, we denotes the decoder current hidden state as h_t and the hidden states from the last layer of the encoder (output of equation (4.1)) as b_t. Hence, c_t which is a weighted sum of b_i is computed as follows:

$$c_t = \sum_{i=1}^{N} a_t(i) b_i \tag{4.2}$$

where a_t is a variable-length alignment vector, and in our case, is based on the global attention of (Luong et al., 2015). In the global attention, the decoder pays attention to all of the hidden states of the encoder. The size of a_t is equal to the number of time steps on the input side and is computed by a comparison between the hidden state of the encoder b_i and the current hidden state of the decoder h_t:

$$\begin{aligned} a_t(i) = \text{align}(h_t, b_i) &= \text{softmax}(\text{score}(h_t, b_i)) \\ &= \frac{\exp(\text{score}(h_t, b_i))}{\sum_j exp(\text{score}(h_t, b_j))} \end{aligned} \tag{4.3}$$

The value of $a_t(i)$ represents the effect of the word i in predicting the next word in the output question. Luong et al. (2015) proposed three different score functions for the alignment vector:

$$\text{score}(h_t, b_i) = \begin{cases} h_t^T b_i & dot \\ h_t^T W_a b_i & general \\ v_a^T \tanh(W_a[h_t; b_i]) & concat \end{cases}$$

After using all the score functions and comparing their results according to the evaluation methods explained in the next chapter, we decided to apply the *general* score as we found it to perform the best in our configuration.

The decoder predicts the next word q_t given the context vector c_t and all the previously predicted words $\{q_1, ..., q_{t-1}\}$. The process stops when the end-of-sentence token is generated. We use a softmax layer to produce the predictive distribution:

$$p(q_t | q_{1:t-1}, A) = softmax(W_s \widetilde{h}_t)$$

where A is the answer sentence and $\widetilde{h}_t$ is the attentional hidden state which is calculated

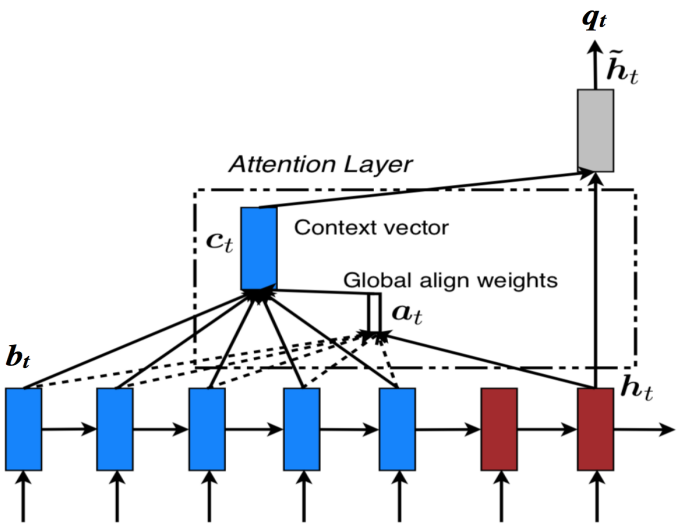


Figure 4.1: Global attentional model (Luong et al., 2015)

given the target hidden state h_t and the source context vector c_t:

$$\widetilde{h}_t = \tanh(W_c[c_t; h_t])$$

The matrices W_s and W_c are weight matrices which will be learned during the training. The hidden state at time step t of the decoder which is a LSTM is generated by:

$$h_t = LSTM(e(q_{t-1}), h_{t-1})$$

where q_{t-1} is the previously generated word and h_{t-1} is the former hidden state.

Figure 4.1 illustrates the global attention mechanism from (Luong et al., 2015).

4.3.3 Coverage Mechanism

One of the problems that many sequence to sequence models suffer from is the repetition problem. This may occur when some words are unnecessarily generated two or more times during decoding. There are also situations where specific words need to be generated, but this mistakenly will not happen. Consider the example below where the input is taken from the Amazon question/answer dataset (Wan and McAuley, 2016):

Answer Sentence: Yes, this rack works great with any size bike as well as boys and girls bikes. I haul my girl's bike and 2 of my grandson's bikes with ease. I love the durability of this frame. The velcro straps for holding the tires in place could be better, so I replaced mine with heavier duty straps, but overall I love it.

Generated Question: Will this rack work with a trek bike bike?

The word *bike* is unnecessarily generated twice. Suzuki and Nagata (2017) dealt with the repetition problem by estimating the upper-bound frequency of the target dictionary while encoding and utilizing that estimation for controlling the output words during decoding. Instead, we address this problem by employing the coverage mechanism. It has been applied recently in (See et al., 2017) for text Summarization. Coverage has been also used in previous research for neural machine translation (Tu et al., 2016; Mi et al., 2016), document summarization (Chen et al., 2016) and image captioning (Xu et al., 2015).

Coverage mechanism keeps track of the number of times a target word is generated and prevents creating the same words over and over again. In this technique, the decoder keeps a coverage vector c^*, which is the sum of the previous alignment vectors:

$$c_t^* = \sum_{t\prime=0}^{t-1} a_t\prime$$

It calculates how much coverage each input word has received from the attention mechanism so far and it helps the mechanism to avoid attending to the same words again once they have been attended to initially (See et al., 2017).

It should be mentioned that c_0^* is a zero vector since nothing has been covered on the first time step. We add this coverage vector to the source hidden states. Hence, we need to update the encoder hidden state b_i as follows:

$$b_i = \tanh(b_i + w_{c^*} c_t^*(i))$$

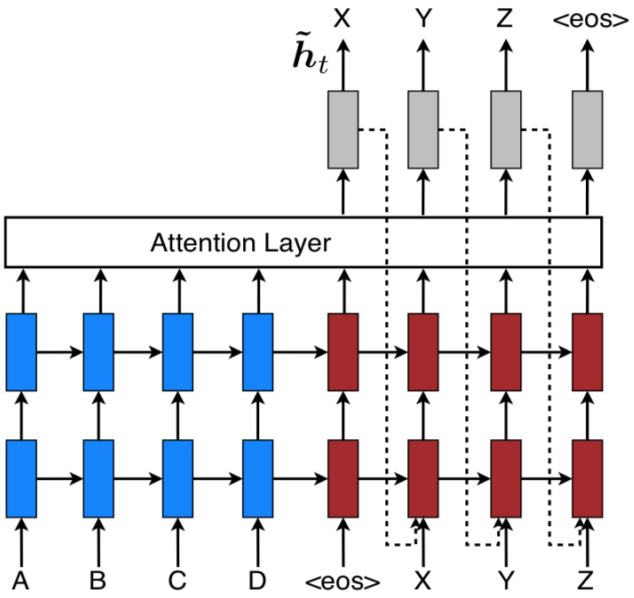


Figure 4.2: Input-feeding approach to make the model aware of the past alignment decisions (Luong et al., 2015).

This b_i will be substituted in equations (4.2) and (4.3) where W_{c^*} is a parameter to be learned. In this way, with the help of c_t^*, the attention mechanism always has a memory of its past decisions.

4.3.4 Input-feeding Approach

In the proposed model, attentional decisions are made independently without taking into account past alignment information. To make the model aware of the past alignment choices, we apply an input-feeding approach from (Luong et al., 2015). This technique informs the decoder what words were considered for the past alignments. We do this by concatenating the attentional hidden state $\widetilde{h}_t$ with inputs at the next time steps. The model spans both horizontally and vertically leading to a deeper network. Figure 4.2 illustrates the concept of input-feeding approach and the figure **??** shows the

4.4 Training and Generation

The goal of the training is to minimize the negative log-likelihood of the training corpus with respect to all model parameters. Consider the corpus consisted of answer and question pairs as $S = \{(a_i, q_i)\}_1^{|S|}$. We define the objective as:

$$J_t = \sum_{i=1}^{|S|} -\log p(q_i|a_i;\theta), \tag{4.4}$$

where θ represents the model parameters. For training our model. we use a strategy called teacher forcing, which means using the exact word from the original question when feeding the next target word to the decoder rather than the decoder's own prediction. This is because of the fact that in the early stages of training, decoder makes many mistakes during generation, and feeding those mistakes as the next input to the decoder can make the training process harder and longer. However, during testing, we feed the model's own output as the next target word to the decoder. This technique prevents the model from learning from mistakes in the process.

In addition to this primary loss function, a coverage loss function is required to penalize an overlap between the coverage vector and the attention distribution, which means attending to the same location multiple times:

$$covloss_t = \sum_i \min(a_t(i), c_t^*(i))$$

After being reweigted by some hyperparameter λ, this amount is added to the equation (4.4):

$$J_t = \sum_{i=1}^{|S|} -\log p(q_i|a_i) + \lambda covloss_t \tag{4.5}$$

As mentioned in section 4.3.2, we do not set a fixed-length vectors for our input sentences. Instead, we add a special end-of-sentence token <EOS> to our dictionary of words which helps with learning variable length sentences. During training, this token is added to

the end of each sentence and while testing the model continues decoding until it generates the <EOS> token (Bengio et al., 2015).

In the generation process, for producing the next word, the decoder can either choose the most likely word according to the model or apply a beam search to generate k best sequences. In our case, a beam search is utilized for the inference.

4.5 Out of Vocabulary Problem

The number of unique words in our dataset reaches about 250k. Dealing with this large amount of data is highly problematic for sequence-to-sequence models as it can increase the training time. Particularly, a softmax layer is used to predict the next target word using a probabilistic distribution over the entire target vocabulary, and the computations can be quite expensive. The computation can also grow proportionally to the vocabulary size. To address this problem, we reduce the vocabulary size on both source and target side. Specifically, we set the vocabulary to the 50k most frequent words on each side, and the rest of the words are replaced with the unknown (*UNK*) token. Reducing the vocabulary size speeds up the training and testing process.

Since the size of our vocabulary is limited to a small number, many *UNK* tokens will be generated during the inference. There are a few methods proposed to address this problem. Gulcehre et al. (2016) deals with this issue in neural machine translation and text summarization task by copying from the source sentence. With the help of an attention mechanism, their model learns where to point in the source sentence, and by utilizing the decoder's hidden state, a binary variable is generated to indicate when to copy from the source sentence. According to the value of this binary variable, the final output can be either a word generated based on a softmax decision over a candidate list as before, or a word that has been copied from the source text. Gu et al. (2016) propose their model called *COPYNET*, which is a mechanism that places a specific part of the source sentence into the proper location in the target sentence.

In our work, to handle this problem, we substitute the *UNK* token at time step t with the word with the highest attention weight from the source sentence. We calculate the index of this word using:

$$\underset{i}{\operatorname{argmax}}\, a_t(i)$$

4.6 Summary

In this chapter, we presented the main motivation behind developing the opinion question generation system and also described other reasons behind preceding studies on the question generation. We formulated our problem and demonstrated the structure of our system as a sequence to sequence model using an encoder and attention-based decoder. The mechanism that we use is the global attention with a general score function from (Luong et al., 2015). For preventing the model from producing repetitive words, we applied a coverage mechanism of (See et al., 2017). In addition, an input-feeding approach was utilized to consider past alignment decisions during training. We introduced our loss function and described training and inference processes. Finally, we elaborated on our proposed solution to the problem of out of vocabulary words. In the next chapter we present the evaluation methods for comparing the performance of our system with another model.

Chapter 5

Experiments and Discussion

In this chapter we present additional details regarding the implementation of our system, including details of the dataset. We also present the baseline, which we used for comparison. We also introduce all of the automatic evaluation metrics that we utilized to assess the results. Lastly, we provide a thorough analysis of our system and the results.

5.1 Dataset

For evaluating opinion question generation. we utilized a community question answering dataset where people can post their questions, and other users can answer them. Specifically, we use the Amazon question/answer dataset (Wan and McAuley, 2016) which contains around 1.4 million question and answer data on 191 thousand products from Amazon. The dataset contains 21 product categories. We select eight categories in order to ensure a diverse dataset. The statistics for each category are shown in Table 5.1.

We replace all URLs in the dataset with the *URL* token to reduce the vocabulary size. We make all the words in our corpus lowercase to reduce the vocabulary size. Otherwise *The* and *the* would be considered as two different words by the model, which is not desirable. Furthermore, we use the NLTK toolkit [6] for sentence tokenization as described in section 5.2.

We set the minimum length of questions to four tokens, including the question mark to filter out poorly structured questions. There can be many examples where the questions are

[6] http://www.nltk.org

Table 5.1: Statistics for each category in the Amazon question/answer dataset.

Category	# questions
Baby	28,933
Electronics	314,263
Grocery and Gourmet Food	19,538
Health and Personal Care	80,496
Home and Kitchen	184,439
Pet Supplies	36,607
Sports and Outdoors	146,891
Tools and Home Improvement	101,088

not grammatically correct; for example, people might just ask: "Waterproof ?". Answers must also contain at least ten tokens, as the same problem can occur here; for example, the answer might be a single 'Yes'. These kinds of answers make the training hard as they do not contain enough useful information. Similarly, we set the maximum length of questions and answers to 20 and 35 tokens respectively. We shuffle all the categories through the dataset and Since all of the examples within a batch which we explain in section 5.3, should also come from different parts of the dataset, we shuffle all the samples before training. Batches are also randomly selected during training by using a function that returns a random permutation of integers from 0 to (*size of the input*) - 1.

We use 80% of the dataset as the training set and the rest is divided between the validation set and the test set. The validation set is applied to assess the convergence of the training. Table 5.2 shows the total number of examples in each dataset after removing very long or very short sentences in only the training set and the validation set. We can not perform any preprocessing or modify the length of the sentences during testing, and because of this, the number of examples in the testing set is more than the validation set.

5.2 Tokenization

Tokenization is the process of breaking up a text into smaller units called tokens. This token can be either a sentence or a single word, punctuation mark or number. Hence,

Table 5.2: Statistics of the processed dataset

	Train	Validation	Test
# pairs	233729	28969	70648

there are two types of tokenization: sentence tokenization and word tokenization. Sentence tokenization is performed on a given paragraph splitting it into a set of sentences. In our work, we set the maximum length of the answer sentences to 35 words. Usually, this number of words constructs two or three sentences. However, the majority of the questions in our dataset are only one sentence. Therefore, we only perform word tokenization on the entire dataset. The reason behind this process is to find all the unique words in our corpus in order to build the dictionary or vocabulary. Later, we assign an index to each of these words to make them a distinctive member of the dictionary.

Word tokenization is the process of splitting a given sentence into single words. This is done by finding the word boundaries, the ending point of a word and beginning of the next one. For performing this task, we use the NLTK toolkit. Here is an example of word tokenization:

Input Sentence: Bob dropped the apple. Where is the apple?

Output tokenization: ['Bob', 'dropped', 'the', 'apple', '.', 'Where', 'is', 'the', 'apple', '?'].

5.3 Implementation details

We implemented our model using the PyTorch[7] package that is integrated into Python. PyTorch is a deep learning framework that provides maximum flexibility and speed. It includes Deep Neural Networks and tensor (n-dimensional array) computation with robust GPU acceleration. Tensors can be placed on either the CPU or GPU. PyTorch uses the memory much more efficiently compared to other alternatives. PyTorch accelerates training

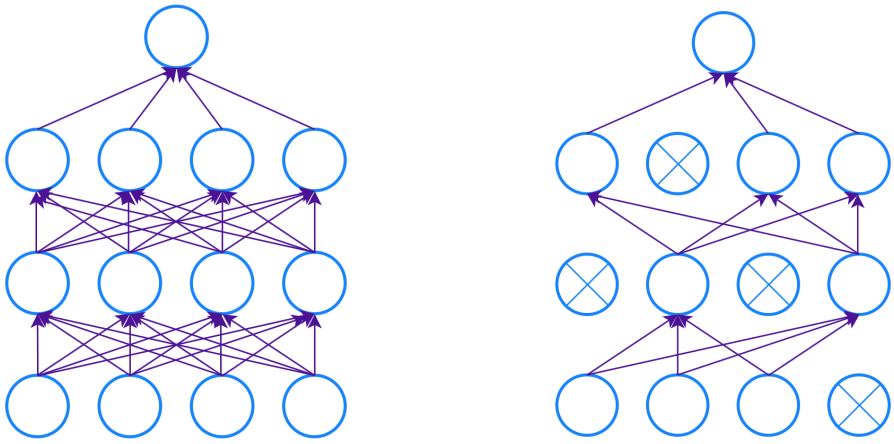
[7] http://pytorch.org

bigger models with its custom memory allocators for a GPU. For training our model, we used a Nvidia TITAN X GPU card with 12GB of RAM.

One of the first steps before training is building our source and target vocabulary (or dictionary). We do this by finding all the unique words on source and target side and assigning an index to each of them. We fix the size of both the source and target vocabularies to 50k. Only the most frequent words are kept on each vocabulary, and the rest are replaced with the *UNK* token. Other tokens that we utilize in our system are *BOS*, which is added to the beginning of a sentence and *EOS*, which is added to the end of a sentence. *EOS* token help with defining variable-length sequences in our model and indicates when the generation of questions should stop. We set word the embedding vector dimension to 300 and use *glove.840B.300d*[8] (Pennington et al., 2014) as our pre-trained word embedding on both the encoder and decoder side. This means we initialize our embedding layer with the weights from this model. If there is any word in our vocabulary that does not exist in the glove word embedding, that word's weight is initialized randomly from a uniform distribution. The word vectors are updated during training and are fine-tuned for our task. The *glove.840B.300d* embeddings contains 840 billion tokens of web data that has been collected from Common Crawl[9], and the dimensions of the vectors are 300.

As we mentioned before, We use a LSTM architecture with a hidden unit size of 600 for our model and set the number of layers to 2 on both the encoder and decoder side. Model parameters are initialized over a uniform distribution. We employ stochastic gradient descent (SGD) as our optimization method with the initial learning rate of 1.0. SGD is a useful approach when working with a huge dataset. It updates the model parameters using only a few training samples or a minibatch instead of just a single example, and the learning rate indicates the rate that we use to update our model parameters. We start halving the learning rate after ten epochs and at the end of each remaining epochs. The training continues for 20 epochs. One epoch is completed when the model tries all the

[8]https://nlp.stanford.edu/projects/glove/
[9]http://commoncrawl.org



(a) Standard neural network without any dropout

(b) Neural network after applying dropout

Figure 5.1: (a) Shows a standard neural network with two hidden layers. (b) The same network after applying dropout to its units (Srivastava et al., 2014).

samples of the dataset. We use a batch size of 64, which represents the number of samples that go through one forward or backward pass. If we choose a big number for the batch size, more memory space will be used. Suppose we have 1000 training examples and the batch size is 100. Therefore, we need ten passes to complete one epoch.

In addition, we use dropout with a probability of 0.3 between vertical LSTM stacks. Dropout is a regularization technique introduced by Srivastava et al. (2014) to prevent the model from overfitting, which happens when the model fits the current dataset very well but fails to generalize to new examples. Dropout is performed by dropping out units from the network along with their connections during training. Choosing which units to drop is random. Figure 5.1 presents a simple neural network before and after applying dropout to its units.

Furthermore, the hyperparameter λ in Equation 4.5 that is used for weighting the coverage loss is set to 1. We also experimented with $\lambda = 2$ but did not find this to be helpful. Decoding is done using a beam search with the beam size of 5 and we stop when we reach the *EOS* token. At the end, we choose the model with the lowest perplexity on the validation set, which shows how well our model fits the unseen data. Perplexity is defined as:

$$PPl(X,Y) = exp(\frac{-\sum_{i=1}^{|Y|} \log P(y_i|y_{i-1},\ldots,y_1,X)}{|Y|}),$$

where X represents the source sentence and Y is the target sentence. The numerator shows the negative log likelihood and the loss function value. The model fits the training data well when the perplexity decreases through the training process.

5.4 Baseline

We are going to compare our model to that of Du et al. (2017). They study the task of question generation in the reading comprehension task. Their model works on both sentence and paragraph level. They utilized an attentional sequence to sequence model with the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). SQuAD is a reading comprehension dataset which includes 536 Wikipedia articles with 100k questions posed on them by Amazon Mechanical Turks crowd-workers, where the answer to each question is a part of the article text.

We only experiment with their sentence-level model, which converts a sentence from a text passage to its corresponding question. We run the same Amazon question and answer dataset on the system provided by the first author, which is implemented in Torch[10]. We keep the source and target vocabulary size same as ours, (i.e., 50k). We also change the maximum length of the source and target sentences to 35 and 20 respectively. The minimum length of input and output sentences are changed accordingly as well. Everything else is left at the default values.

5.5 Automatic Evaluation Metrics

For assessing the effectiveness of our system, we are going to evaluate it using three different automatic evaluation metrics, BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and ROUGE-L (Lin, 2004). The package that we apply for these metrics

[10]http://torch.ch

is by Chen et al. (2015). In the following sections, we are going to describe the necessary background regarding these metrics.

5.5.1 BLEU

BLEU (Bilingual Evaluation Understudy) (Papineni et al., 2002) was first introduced for the machine translation task. Its objective is to reduce the required time and effort of human labour in evaluating the translations. It is also language-independent, inexpensive, quick and has a high correlation with human judgment.

BLEU calculates the N-gram overlap between the hypothesis and reference sentence and considers a penalty for very short sentences; however, it does not consider synonymy or paraphrasing. In our work, we only consider BLEU-1 and BLEU-2, which calculates the overlap between unigrams and bigrams.

BLEU's output varies from 0 to 1, with the values closer to 1 showing more similar results to the reference. However, reaching the value one means the result is identical to the reference, and this is not possible in practice. For instance, suppose we ask two different persons to translate a sentence from French to English. Their translations will not be identical to each other and the reference. Therefore the scores would be different and below 1.

5.5.2 METEOR

If we consider the machine translation task, METEOR (Metric for Evaluation of Translation with Explicit ORdering) (Denkowski and Lavie, 2014) scores predictions by aligning them to ground truth sentences and measuring the sentence-level similarity scores. It performs these alignments with the help of exact word matching, stemming, synonyms and paraphrases. METEOR employs recall and precision for performing the matching.

5.5.3 ROUGE

ROUGE, or Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004), is used for evaluating text summarization and machine translation. It compares the generated summary or translation with the references based on the n-gram.

There are five variations of ROUGE metric: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. ROUGE-N computes the overlap of N-grams, for instance, ROUGE-2 calculates the overlap of bigrams between the hypothesis and the references. ROUGE-L reports the matching results based on the Longest Common Subsequence (LCS). The longest common subsequence of two different sequence X and Y is a common subsequence with the maximum length. ROUGE-W gives higher scores to consecutive LCSes, and it is known as the weighted LCS. Finally, for describing ROUGE-S, we first need to define a Skip-bigram. A skip-bigram refers to any pair of tokens in their sentence order. There can be any arbitrary gaps between them. Consider the sentence "bird on the tree". The skip-bigrams are: {"bird on", "bird the", "bird tree", "on the", "on tree", "the tree"}. ROUGE-S or Skip-Bigram Co-Occurrence Statistics computes the overlap of skip-bigrams between a hypothesis and references.

For evaluating our system, we use ROUGE-L. The longer the LCS of two sequences is, the score is higher. Consider this example from (Lin, 2004):

1. police killed the gunman
2. police kill the gunman
3. the gunman kill police

Consider the first sentence as the reference and the second and third ones as the summary sentences. Calculating the ROUGE-L score is as follows:

$$ROUGE_L = \frac{\text{LCS}(\text{hypothesis}, \text{reference})}{\text{Total number of reference words}}$$

Table 5.3: Comparison of BLEU 1-2, METEOR and $ROUGE_L$ scores between our model and (Du et al., 2017).

	BLEU 1	BLEU 2	METEOR	$ROUGE_L$
(Du et al., 2017) model	12.89	6.95	8.76	**25.91**
Our model	**14.67**	**7.74**	**9.43**	25.21

So second sentence score (S_2) would be $\frac{3}{4} = 0.75$ ("police the gunman") and $S_3 = \frac{2}{4} = 0.5$ ("the gunman"). Hence, from ROUGE-L perspective, S_2 is considered better than S_3.

5.6 Automatic Evaluation Results

The comparison of automatic evaluation metrics between our system and the Du et al. (2017) model is shown in Table 5.3. The bold numbers demonstrate the best performing system for each evaluation metric.

As shown in table 5.3, our model improves the BLEU 1 score by at least 1.5 points. It also achieves a better result regarding BLEU 2 and METEOR, whereas the ROUGE is lower than the baseline. One of the important things worth mentioning is that all scores are quite low in contrast to other works on question generation. If we consider the results reported in the Du et al. (2017) study, we notice that the BLEU scores are much higher compared to our work. The reason lies in the dataset they are using, which is SQuAD dataset (Rajpurkar et al., 2016). SQuAD is a human-generated corpus from Wikipedia articles. The sentences are well-structured, grammatically correct with fewer unnecessary punctuation and colloquialism. However, when working with community-based question answering systems, sometimes the structure of sentences do not follow the correct grammatical and semantical structure. There are also a significant amount of useless information and symbols in the sentences. Most of the sentences resemble an everyday conversation between people. In other words, community-based question answering datasets contain a lot of noise and the Amazon dataset is no exception; therefore training with such dataset can be harder, which affects the results.

Another problem that exists in question generation is that multiple questions can be

Table 5.4: Question and answer samples from the Amazon dataset

Question 1:	Is there a trick to get this to work? The top lever does not move up and down for the water to flow. What am I missing?
Answer 1:	I think yours is defective. I have two and they work fine. No tricks.
Question 2:	Does it work with the Sony Vaio pro 13 touch SVP13213CXB?
Answer 2:	I don't know the answer to that question. Sorry. Wish I could help.

generated from a single sentence. The system might generate a question which is correct both semantically and grammatically and also asks about some accurate information in the sentence. However, it is not the same as the ground-truth and this results in lower scores.

Consider the first example in Table 5.4, which we have picked manually from the Amazon question/answer dataset. The last part of the answer is not a full sentence; it is not stated anywhere what the discussed object is in order to form a question about it. Turning such a sentence into an interrogative form and also generating a question which is close to the real ones in the original sentence can be very challenging considering the available information in the answer. In the second example, the person responding to the question is just stating being unaware of the facts required to answer the question. These kinds of answers are seen quite a lot in the dataset, and many questions can be associated with them.

Therefore, there are many factors which lead to lower results when using automatic evaluation metrics. The problem of evaluating the results of a question generation system due to the low word overlap with the ground-truth sentence exists in other studies as well (Yuan et al., 2017). Hence, applying a corpora with multiple ground-truth questions can help with evaluating the system (Mostafazadeh et al., 2016).

Table 5.6 shows a few examples generated by our system and (Du et al., 2017) (DSC). GT denotes the ground truth questions. As demonstrated, the questions generated by our system have more overlap with the ground truththan the DSC system, which explains the higher BLEU 1 score. However, this does not mean their system does not generate accurate questions. This problem arises due to the mentioned problems with the dataset. For instance, in the first example, the sample generated by the DSC system can be accepted as

an accurate question, as it is pointing out to useful information in the answer. However, our system is closer to the original question which results in higher scores. In addition, one of the advantages of our system over DSC can be realized from the example 4 and 6, where the coverage mechanism becomes useful and prevents the model from generating the same word 'material' in example 4 and 'BP' in example 6 again.

There is still a considerable need for a less noisy dataset for opinion question generation task. There are other community-based question answering datasets such as Yahoo! Answers, Quora, StackOverflow, and StackExchange but none of which provide a clean, less noisy dataset that does not pose the mentioned problems. Improving such datasets or proposing a strategy that can efficiently deal with noise can be an interesting topic of study which will spur research in other areas such as question answering as well.

5.7 Human Evaluations

To further assess the performance of our system, we performed human evaluations on the results of our system and the baseline. Two anonymous English-speaker students judged the questions generated from both systems. They considered two criteria in their judgment: **syntactic correctness** and **relevance**. Syntactic correctness indicates the grammaticality and fluency of the generated questions and relevance demonstrates whether the generated question is meaningful and related to the sentence it is generated from. For instance, considering the sentence "It is cloudy in Lethbridge today.", the question: "How is the weather in Lethbridge?" is considered more relevant than the question "Does Lethbridge have an airport?".

The judges were asked to give a score from 1 (very poor) to 5 (very good) to each system separately according to the mentioned criteria. They performed the evaluations on 100 randomly selected question and answer pairs from the results. The comparison of human evaluations between our system and the Du et al. (2017) model is shown in Table 5.5. Bold numbers demonstrate the best performing system for each evaluation criteria.

Table 5.5: Human evaluation results for syntactic correctness and relevance between our model and (Du et al., 2017).

	Syntactic correctness	Relevance
(Du et al., 2017) model	4.4	2.93
Our model	**4.52**	**3.37**

5.8 Summary

In this chapter, we shed light on the Amazon question/answer dataset and described how we preprocessed it to make the training process easier. We also provided a detailed explanation of the processes we underwent in the implementation of our model. We defined all the automatic evaluation metrics applied for assessing our model. We presented a baseline for comparing our model with and demonstrated the metrics' outcome on both of the systems. Finally, we analyzed some samples from the dataset and the results from both our system and the baseline.

Table 5.6: Examples of the generated questions from DSC (Du et al., 2017) and our model, accompanied by the answers and the ground truth (GT) questions.

Answer 1: Powdered goat milk is still milk and therefore has lactose. **GT Question:** Is this lactose free? **DSC:** what are the ingredients? **Ours:** is this product lactose free?
Answer 2: I really don't know, I did full size cupcakes, mini ones it would hold a ton! **GT Question:** How many mini-cupcakes will this hold? **DSC:** what size is it? **Ours:** how many cupcakes will it hold?
Answer 3: 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? **DSC:** What is the assembly required? **Ours:** what tools do I need to assemble this?
Answer 4: 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? **DSC:** Can you still use the material while wearing the material? **Ours:** Can I do pushups while wearing these wraps?
Answer 5: 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? **DSC:** 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 6: 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? **DSC:** What is the difference between the BP and the BP? **Ours:** How do you print from the BP?
Answer 7: I haven't used the touch functions, but yes, I got video in windows 7. **GT Question:** This monitor supports windows 7? **DSC:** Does it have a built in Webcam? **Ours:** Does it work with windows 7?
Answer 8: Lightweight. It can be lifted easily when folded up. **GT Question:** What is the weight? **DSC:** Is it hard to fold up? **Ours:** How much does it weigh?

Chapter 6

Conclusion

6.1 Summary and Conclusion

In this thesis, we address the opinion question (QG) generation problem. QG, in general, has a profound impact on people's everyday life. Students can benefit from such system to test their knowledge when preparing for an exam or when they are self-studying. QG can help people to realize their knowledge deficits and encourages them to seek more information to make up for those deficiencies. Teachers can utilize QG to present challenging questions to inspire students during lectures or come up with new ideas when preparing exam material. Medical chatbots and other intelligent personal assistants can be further improved with QG. It also helps in enhancing other research areas in natural language processing such as question answering and the reading comprehension task by providing a better dataset for them. Specifically for our task, we develop an opinion question generation system to aid the search engines in offering a couple of sample questions to the users when they enter a query. The reason behind this is that sometimes people may not enter their exact, desired question, and therefore obtaining many unwanted results. For evaluating this system, we utilized community-based question answering systems which are close to people's thoughts and everyday speech. To be more specific, we utilized the Amazon question/answer dataset.

Based on the successful results of the sequence-to-sequence approach in neural machine translation, text summarization, image caption generation and question answering, we are applying the encoder-decoder framework to our model. When people try to form

a question from a sentence, they usually focus on a specific part of a sentence and create a question on that part. Inspired by the same technique, we employ the attention mechanism to make the model pay attention to a particular part of the input sentence. We use the global attention technique from Luong et al. (2015). To further boost the performance of our system we adopt a coverage mechanism, which is one of the solutions to the repetition problem. Many sequence-to-sequence models experience the repetition problem. However, a coverage mechanism prevents this by keeping track of what has been generated so far. Another technique that we apply to the advancement of our model is an input-feeding approach which informs the decoder what words were considered for the past alignments. In addition, due to the limited size of output vocabulary, many unknown tokens will be generated. We proposed our solution to this problem in Chapter 4. The detailed structure of the encoder, attentional decoder, and other mentioned techniques are also discussed in this chapter. We define our task precisely and present the training and generation procedures.

In Chapter 2, we briefly discussed the intuition behind the deep learning approach, and we presented all the essential backgrounds that are required for understanding the structure of our system. Later in Chapter 3, we first introduced the traditional methods utilized to address the QG problem. Afterwards, we described recent sequence-to-sequence models followed by other successful studies in natural language community on encoder-decoder models.

In chapter 5, we elaborated on the dataset and presented the necessary background for preprocessing the inputs. We provided implementation details and introduced the baseline model. We also employed three different evaluation metrics: BLEU, METEOR, and ROUGE and presented our results based on these metrics. Finally, we provided a discussion according to the generated outputs of our system and the baseline.

We mentioned two problems that resulted in low scores in the automatic evaluation metrics and addressing these problems requires more work in the QG research topic. We plan to tackle these problems with the directions explained in the next section.

6.2 Future Work

We can extend the proposed model and further improve its performance according to these directions:

- We can change the structure of our encoder and decoder from a bidirectional RNN and experiment with a transformer model (Vaswani et al., 2017) which is based only on an attention mechanism and does not use any recurrence.

- As discussed earlier, one of the problems with evaluating results of a QG system with automatic metrics is that we can generate multiple questions from a single sentence, all questions can be grammatically correct and ask about proper information in the input sentence. However, if they have low word overlap with the ground-truth the results would be considered insignificant. For facilitating the evaluation process and obtaining more accurate results, we can use a dataset which consists of one sentence with multiple questions on different parts of the input.

- One further step for improving the dataset involves more preprocessing. For instance, we can apply the technique in Du and Cardie (2017) which is finding the question worthy sentences in the text and form questions only on those sentences.

References

O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. 2014. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22(10):1533–1545.

Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pages 56–64.

Manish Agarwal, Rakshit Shah, and Prashanth Mannem. 2011. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pages 1–9.

H. Ali, Y. Chali, and S. A. Hasan. 2010. Automation of Question Generation from Sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*. Pittsburgh, USA.

Christof Angermueller, Tanel Prnamaa, Leopold Parts, and Oliver Stegle. 2016. Deep learning for computational biology. *Molecular Systems Biology* 12(7):878–n/a. 878. https://doi.org/10.15252/msb.20156651.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations Workshop (ICLR)* http://arxiv.org/abs/1409.0473.

Lee Becker, Sumit Basu, and Lucy Vanderwende. 2012. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 742–751. http://aclanthology.coli.uni-saarland.de/pdf/N/N12/N12-1092.pdf.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, NIPS'15, pages 1171–1179. http://dl.acm.org/citation.cfm?id=2969239.2969370.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and Trends in Machine Learning* 2(1):1–127. https://doi.org/10.1561/2200000006.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3:1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Mohan John Blooma and Jayan Chirayath Kurian. 2011. Research issues in community based question answering. In *Pacific Asia Conference on Information Systems, PACIS 2011: Quality Research in Pacific Asia, Brisbane, Queensland, Australia, 7-11 July 2011*. page 29.

Denny Britz. 2015. Recurrent neural networks tutorial, part 1 introduction to rnns. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 819–826.

Yllias Chali and Sina Golestanirad. 2016. Ranking automatically generated questions using common human queries. In *Proceedings of the 9th International Natural Language Generation Conference*. Edinburgh, Scotland.

Yllias Chali and Sadid A. Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics* 41(1):1–20. https://doi.org/10.1162/COLI_a_00206.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2015)*. AAAI, New York, NY.

Wei Chen, Gregory Aist, and Jack Mostow. 2009. Generating questions automatically from informational text. In *Proceedings of the 2nd Workshop on Question Generation*. pages 17–24.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint* arXiv:1504.00325. http://arxiv.org/abs/1504.00325.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry nau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1724–1734. https://doi.org/10.3115/v1/D14-1179.

Junyoung Chung, Çalar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv e-prints* abs/1412.3555. Presented at the Deep Learning workshop at NIPS2014. https://arxiv.org/abs/1412.3555.

Olivier Delalleau and Yoshua Bengio. 2011. Shallow vs. deep sum-product networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, Curran Associates, Inc., pages 666–674. http://papers.nips.cc/paper/4350-shallow-vs-deep-sum-product-networks.pdf.

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

Xinya Du and Claire Cardie. 2017. Identifying where to focus in reading comprehension for neural question generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2057–2063. http://aclweb.org/anthology/D17-1218.

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

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, USA, CVPR '14, pages 580–587. http://dx.doi.org/10.1109/CVPR.2014.81.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Arthur C. Graesser and Natalie K. Person. 1994. Question asking during tutoring. *American Educational Research Journal* 31(1):104–137. http://dx.doi.org/10.3102/00028312031001104.

A. Graves, A. r. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. pages 6645–6649.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint* arXiv:1308.0850. https://arxiv.org/abs/1308.0850.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. JMLR.org, ICML'14, pages II–1764–II–1772. http://dl.acm.org/citation.cfm?id=3044805.3045089.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1631–1640. https://doi.org/10.18653/v1/P16-1154.

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

Sadid A. Hasan. 2013. *Complex Question Answering: Minimizing the Gaps and Beyond*. Ph.D. thesis, University of Lethbridge.

Michael Heilman and Noah A. Smith. 2009. Question generation via overgenerating transformations and ranking.

Michael Heilman and Noah A Smith. 2010a. Extracting simplified statements for factual question generation. In *Proceedings of QG2010: The Third Workshop on Question Generation*. pages 11–20.

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

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 2073–2083. https://doi.org/10.18653/v1/P16-1195.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*. ACM, New York, NY, MM '14, pages 675–678. http://doi.acm.org/10.1145/2647868.2654889.

Saidalavi Kalady, Ajeesh Elikkottil, and Rajarshi Das. 2010. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*. pages 1–10.

Andrej Karpathy and Li Fei-Fei. 2017. Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(4):664–676. https://doi.org/10.1109/TPAMI.2016.2598339.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pages 1097–1105.

Girish Kumar, Rafael Banchs, and Luis Fernando D'Haro. 2015. Revup: Automatic gap-fill question generation from educational texts. In *10th Workshop on Innovative Use of NLP for Building Educational Applications*. pages 154–161.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 889–898.

Byunghan Lee, Junghwan Baek, Seunghyun Park, and Sungroh Yoon. 2016. deeptarget: End-to-end learning framework for microrna target prediction using deep recurrent neural networks. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, New York, NY, USA, BCB '16, pages 434–442. https://doi.org/10.1145/2975167.2975212.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81. http://aclweb.org/anthology/W/W04/W04- 1013.pdf.

David Lindberg, Fred Popowich, John C. Nesbit, and Philip H. Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Sofia, Bulgaria, pages 105–114. http://www.aclweb.org/anthology/W13-2114.

Ming Liu, Rafael A Calvo, and Vasile Rus. 2010. Automatic question generation for literature review writing support. In *International Conference on Intelligent Tutoring Systems*. Springer, pages 45–54.

Minh-Thang Luong. 2016. *Neural Machine Translation*. Ph.D. thesis, Stanford University.

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

Karen Mazidi and Rodney D. Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 321–326. http://aclanthology.coli.uni-saarland.de/pdf/P/P14/P14-2053.pdf.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 955–960. https://doi.org/10.18653/v1/D16-1096.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Curran Associates, Inc., NIPS'13, pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 746–751. http://aclanthology.coli.uni-saarland.de/pdf/N/N13/N13-1090.pdf.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing - Volume 2*. Association for Computational Linguistics, pages 17–22.

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

J. Mostow and W. Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling*. pages 465–472.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint* arXiv:1703.01619. https://arxiv.org/abs/1703.01619.

Quoc Bao Nguyen, Tat Thang Vu, and Chi Mai Luong. 2015. *Improving Acoustic Model for Vietnamese Large Vocabulary Continuous Speech Recognition System Using Deep Bottleneck Features*, Springer International Publishing, Cham, pages 49–60. https://doi.org/10.1007/978-3-319-11680-8$_5$.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint* arXiv:1611.09268. http://arxiv.org/abs/1611.09268.

E. Nishani and B. io. 2017. Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation. In *2017 6th Mediterranean Conference on Embedded Computing (MECO)*. pages 1–4.

Chris Olah. 2014. Deep learning, nlp, and representations. http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/.

Chris Olah. 2015. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. http://aclanthology.coli.uni-saarland.de/pdf/P/P02/P02-1040.pdf.

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

Paul Piwek and Svetlana Stoyanchev. 2010. Question generation in the coda project .

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2383–2392. https://doi.org/10.18653/v1/D16-1264.

Oleg Rokhlenko and Idan Szpektor. 2013. Generating synthetic comparable questions for news articles. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 742–751. http://aclanthology.coli.uni-saarland.de/pdf/P/P13/P13-1073.pdf.

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

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Ivan A. Sag and Dan Flickinger. 2008. Generating questions with deep reversible grammars. In *Proceedings of the First Workshop on the Question Generation Shared Task and Evaluation Challenge*.

Máximo E. Sánchez-Gutiérrez, E. Marcelo Albornoz, Fabiola Martinez-Licona, H. Leonardo Rufiner, and John Goddard. 2014. *Deep Learning for Emotional Speech Recognition*, Springer International Publishing, Cham, pages 311–320. https://doi.org/10.1007/978-3-319-07491-$7_3$2.

Mike Schuster, Kuldip K. Paliwal, and A. General. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* .

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

Iulian V. Serban, Alberto García-Durán, Çalar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016a. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 588–598.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pages 3776–3783.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016c. A hierarchical latent variable encoder-decoder model for generating dialogues http://arxiv.org/abs/1605.06069.

Richard Socher. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Ph.D. thesis, Stanford University.

Linfeng Song, Zhiguo Wang, and Wael Hemza. 2017. A unified query-based generative model for question generation and question answering. *arXiv preprint* arXiv:1709.01058. https://arxiv.org/abs/1709.01058.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pages 553–562.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, NIPS'14, pages 3104–3112. http://dl.acm.org/citation.cfm?id=2969033.2969173.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 291–297. http://aclanthology.coli.uni-saarland.de/pdf/E/E17/E17-2047.pdf.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*. http://arxiv.org/abs/1409.4842.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint* arXiv:1706.02027. http://arxiv.org/abs/1706.02027.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 76–85. https://doi.org/10.18653/v1/P16-1008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint* arXiv:1706.03762. http://arxiv.org/abs/1706.03762.

Mengting Wan and Julian McAuley. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *Proceedings of the International Conference on Data Mining (ICDM)*. pages 489–498.

Weiming Wang, Tianyong Hao, and Wenyin Liu. 2007. Automatic question generation for learning evaluation in medicine. In *Proceedings of the 6th International Conference on Advances in Web Based Learning*. Springer, pages 242–251.

Brendan Wyse and Paul Piwek. 2009. Generating questions from openlearn study units. In *AIED 2009 Workshop Proceedings Volume 1: The 2nd Workshop on Question Generation*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Lille, France, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1040–1050. http://aclweb.org/anthology/P17-1096.

Xuchen Yao and Yi Zhang. 2010. Question generation with minimal recursion semantics. In *Proceedings of QG2010: The Third Workshop on Question Generation*. Citeseer, pages 68–75.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4:259–272.

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

Xin Zhen, Jiawei Chen, Zichun Zhong, Brian Hrycushko, Linghong Zhou, Steve Jiang, Kevin Albuquerque, and Xuejun Gu. 2017. Deep convolutional neural network with transfer learning for rectum toxicity prediction in cervical cancer radiotherapy: a feasibility study. *Physics in Medicine Biology* 62(21):8246. http://stacks.iop.org/0031-9155/62/i=21/a=8246.

Zhicheng Zheng, Xiance Si, Edward Y. Chang, and Xiaoyan Zhu. 2011. K2q: Generating natural language questions from keywords with user refinements. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*. page 947955.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint* arXiv:1704.01792. http://arxiv.org/abs/1704.01792.

Appendix A

Samples of the Generated Questions

Generated Questions

In this section, we demonstrate some of the generated questions from our system described in chapter 4 along with the input answer and the ground truth questions.

Answer: This is a normal soccer sock thickness. It is not padded.
Original Question: Are the socks thick? I am looking for a thicker sock.
System Generated Question: are the socks thick? I am looking for a thicker sock.

Answer: This lens is compatible with all Sony crop sensor DSLRs. So yes it will work with your camera . It's a great budget prime lens.
Original Question: I'm using the a37, so is this compatible with my camera? Thanks in advance!
System Generated Question: Is this lens compatible with the canon rebel t3i ?

Answer: Yes, this permanent filter will fit model jwx5.
Original Question: Does this fit in the Mr Coffee 5 Cup Model #JWX5 Coffee Maker ?
System Generated Question: Will this filter fit model #JWX5?

Answer: Camera, battery, charger (which is in 2 pieces), some cable, the removable flash, the lens, the flash bag, camera body cap, lens cap and strap plus the cd which in cludes editing software. I've never seen a decent camera that comes with a card.
Original Question: What is included when you purchase this camera and lens? Does it include the battery, charger, cables, cards, flashes, etc?
System Generated Question: What comes in the box with the camera?

Answer: I ordered extra bulbs when I ordered the lantern. They sent me rayovac 6v k13/kpr113. I would guess you can get these bulbs at any retailer that sells bulbs. I hope this helps you. Regards.
Original Question: What kind of bulb does it take and where do you get it?
System Generated Question: Where can i get replacement bulbs for this lantern?

Answer: Hi no this laptop does not came with Microsoft office you must install it.
Original Question: Does this laptop come with Microsoft office?
System Generated Question: Does this laptop come with Microsoft office?

Answer: Update on my previous response. The company confirms made in china.
Original Question: Where was it manufactured?
System Generated Question: What is the country of origin of this product?

Answer: I did not need to use an extension for my install and I have granite counter. Make sure the thickness of your counter to see if it matches.
Original Question: Does this really need extension for granite counter install?
System Generated Question: Do you need an extension for granite counter top?

Answer: It uses two uv bulbs with an estimated lifespan of 2+ years.
Original Question: Hi, does this use one, or two uv bulbs? Thanks
System Generated Question: What kind of bulbs does it use?

Answer: It is about 17 1/2 inches square and about 16 1/4 inches high.
Original Question: Can anyone tell me the size? x? And height.
System Generated Question: What are the dimensions of this item?

Answer: Vertical antennas generally need base radials. With a mobile mount such as this one the metal roof of the vehicle supplies that . if you have a house with a metal roof it should work fine . my mag base antenna works fine on the metal roof of my porch , if i brin g it inside and set it on the wooden floor it does n't work well at all .
Original Question: Can i use this with a base station?
System Generated Question: Does the antenna need to be installed on a metal roof or does it need a base?

Answer: When i was researching this type of dog door a website said just to measure your dogs chest and/or hips. Just to make sure they are not wider than the opening. I have a chunky dog so I was skeptical about ordering it but I'm glad I did!
Original Question: What's the largest size dog that this door will accommodate?
System Generated Question: what is the size of the dog door?

Answer: Small or 60 I am 5 "8 and weigh 160 and got medium.
Original Question: I'm 5 ft 7 inches tall and weigh 125 lbs. What size should I order?
System Generated Question: I am 5'7" and weigh 125 lbs. What size should I order?