A Study of Reinforcement Learning for Abstractive Summarization

LE Thanh Tung

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
October, 2018

Master's Thesis

A Study of Reinforcement Learning for Abstractive Summarization

1610438 LE Thanh Tung

Supervisor : Nguyen Minh Le
Main Examiner : Nguyen Minh Le
Examiners : Satoshi Tojo
Kiyoaki Shirai
Shinobu Hasegawa
Hiroyuki Iida

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science

August, 2018

Abstract

In the explosion of information, the need for exploration and understanding the semantic features is more and more important and necessary. The long documents, however, contain so much redundant information, which takes our less concentration on the crucial topics. Summarization is the task of catching the critical information from the input documents to create the short and meaningful summary. In common, summarization task is classified into extraction and abstraction. In the reason that the extractive summary created by selecting the original sentences still contains the redundancy, we choose the main topic in this thesis on Abstractive Summarization task, which is more difficult and significant in practice.

Abstractive summarization is the task of interpreting the meaning of the input documents to generate or rephrase the concise and worthwhile summary. However, text understanding is still the obvious challenge in Natural Language Processing. The difficulty comes from the variety and complexity of human language. In the sense of computational processing, the length of the text is also the huge problem. In spite of these difficulties, it is the motivation to design the automatic summarization system which is extremely useful in practice. Thanks to the summarization system's strength, we can save the time and cost of information processing and storage. Even, it is also used in the social text processing to reduce the length of text in many applications.

With the observation of the drawbacks in text understanding and the previous latest solutions, we propose the novel model with the combination of the different networks and techniques. First of all, we present the most popular approach is used in Abstractive Summarization. It is the Sequence to Sequence model based on the development and success of Deep Learning networks. In these models, they use one kind of network like Convolution Neural Network (ConvNet), Recurrent Neural Network (RNN) and so on to embed the content of input document. After, they use another network to generate the summary sequence in the decoding phase. The quality of these model is based on the strength of the internal integrated networks.

This thesis presents our research on Abstractive Summarization and our contributions in this topic: (1) we propose the combination of ConvNet and RNN-based network to extract both the global and local features of sequence from the input; (2) we integrate the bilinear attention mechanism into our decoding phase which reduces the redundancy and replication in the input and the output in generating step; (3) we apply Reinforced Mechanism into our encoder-decoder model with my proposed modification on reward score which is proved its effectiveness in the experiments.

First of all, we propose the novel encoder model by combining two different kinds of neural networks in the same part. With the local features extraction's strength of ConvNet, we add the essential global meaning of sequence to create the high representation of the input. In the experiment results, this combination proved its importance to map

the input sequence into the meaningful vector space.

Secondly, we apply and implement bilinear attention scoring functions in our model. To overcome the drawback of the long documents, we use the intra-temporal attention which is to eliminate the redundancy and emphasize the importance in the input sequence. Simultaneously, we use the intra-decoder attention scoring function to evaluate the effect of the previous words in the generated summary into the current predicting step. It helps us to avoid the replication of words in the summary.

The last but not least, we apply Reinforced Mechanism into our model to get it closer and closer to the real-world object. By adding the environment's feedback through the reward function, our model gets more robustness and practical ability. In this sense, we also propose the new reward function for Abstractive Summarization problem as the average of three different scores in the evaluation.

In the experiment, the dataset we use is CNN/Daily Mail. The characteristics of this dataset are the massive volume of samples and the considerable length of input and output sequences for each sample. With many experiments, we prove that our proposed model is extremely effective and useful. In this sense, it also outperforms the previous works by 1% increment of ROUGE-1 and 0.4% of ROUGE-2 in Abstractive Summarization. It is a positive signal to show the strength of our proposed model in practice.

Keywords: abstractive summarization, reinforced mechanism, bilinear attention, neural encoder-decoder model, combined objective

Acknowledgment

From my heart, I would like to express my appreciation and sincere to my main supervisor - Associative Professor Nguyen Le Minh for the opportunity he gave me two years ago. When I was an undergraduate student, the greatest wish of my life is to study abroad. In that spring, he gave me a valuable chance to do an internship in JAIST. It is the first time I have gone abroad, and unfortunately, I missed the flight and changed the departure airport to Osaka. In my panic and fear in the first time to Japan, he arrived and picked me up at the station at midnight after 24-hour moving from Vietnam to Osaka and returning to Komatsu. His enthusiasm and kindness is something I will never forget in my life. In the other hands, that internship period motivates me to try harder and harder in my work.

As my destiny, I chose JAIST for my master's study. When the first day I come back JAIST, his enthusiasm has no change. Each day I go to the lab is the hard working day with the great and warm support from Nguyen-sensei. It is the motivation of my daily life. In the darkest days of my research, I seem to leave everything behind and give up the research life to come back to Vietnam. He encouraged and supported me to continue and overcome the difficulty of my topic and gave me a lot of advice in both life and work. I want to give my appreciation again to Nguyen-sensei for everything he gave me in my life.

Simultaneously, I also want to thank my colleagues and senior for their kindness and hospitality. Their contribution to each of my seminar is the great help to me to review and improve my research methodology and my current study. Among many great seniors in my lab, I want to send my special thanks to Tien-Huy san. The help he gave me is over the role of the tutor. It has become the close-knit affair for you and me.

Finally, I want to send my appreciation to all the professors who gave me a lot of useful knowledge in my research life. With their help and education, I have enough confidence to continue on the difficult path of research in the future. Especially, I am also grateful to give my impressed thanks to Prof Satoshi Tojo, Associate Professor Kiyoaki Shirai other examiners for their valuable comments and feedbacks to my study. Besides, I also give my gratefulness to JAIST. With completed support from accommodation and research, JAIST becomes the great place for study. The period of living here is the most beautiful memories in my life.

Last but not least, thank all of my fellow and international friends who are always with me, and constitutes my life full of happiness and success at JAIST

Le Thanh Tung,
August, 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, I will give a fundamental background in summarization to show the motivation in this thesis and our contribution to this topic.

1.1 Background

In the explosion of information, the volume of data we need to process is to grow up rapidly, which leads to the demand of human beings with the meaningful and concise paragraph covering the most information in the long documents. This mechanism is so popular in the newspaper as the title and headlines. To take less time, we only need to read through this essential information instead of reading the whole documents as our habitat in the daily life. For this purpose, the need for tools which can extract the critical parts for the extended source datum automatically is more and more necessary and practical.

Summarization is one of the novel and fascinating fields in Natural Language Processing (NLP), which has the great significance in both industry and scientific research. Based on the way of summary's creation, the task is divided into two kinds: extraction and abstraction. In Extractive Summarization, the important sentences are selected and used directly as the expecting summary whose drawback is the massive redundancy among sentences. The second type, Abstractive Summarization, which is our primary concerns in this thesis, is to generate or rephrase the summary through the input's understanding. This mechanism can eliminate the redundancy of information's replication among the selected sentences in extraction mechanism.

In the early days, the previous works focus on rephrasing from the extractive summarization to eliminate the redundant information. In the sense of generation, these techniques are just considered as the improvement of extraction. In recent, most of the approaches are divided into two categories: structured-based and semantic-based methods. With the structure based techniques, the input documents are represented as the new and meaningful organization, which consists of tree-based [2], template-based [5],

rule-based [7], graph-based [6, 24] and so on. Based on these structures, the meaningful words are selected and connected by the templates, rules and the scoring function which are often based on the expert's knowledge. However, the drawback of these systems is that they use a lot of prior expertise and their summary elements only come from the source documents, which leads to the variety and robustness of the practical systems.

Recently, semantic-based approaches based on Natural Language Generation (NLG) mechanism become more and more popular in Abstractive Summarization. The main idea of these techniques is to represent the input documents in the meaningful space which is fed into the generator to get the sequence of words in the summary [15]. The critical problem for text understanding is the length and language's complexity of sequences which leads to the weakness of prior knowledge. With the development of Deep Learning, the recurrent models prove the effectiveness in linguistic processing. In this sense, these systems often include two phases: encoder and decoder phase which are based on the Neural Network models. The role of the encoder phase is to embed the whole sentences into the meaningful space to get the semantic and relational features. This representation is fed into the decoder model to generate the new sentence that covers the content of the source inputs. There are a lot of models used in recent like Recurrent Neural Network (RNN), Convolution Neural Network (CNN), Long Short-term Memory (LSTM) which will be presented in Chapter 2.

1.2 Motivation

Although RNN-based sequence to sequence model and attention-based score for the input has achieved the good result recently, the RNN-based model also exists some important drawbacks where we do not concern the traditional disadvantages which were improved by the variants like Gated Recurrent Unit (GRU), LSTM, and so on. Among the other Neural Network models, Recurrent Neural Network is so suitable for the sequence learning of sentences or documents. However, with their variants, the RNN-based model just embed the short and average length-sized sequence that is not suitable for this problem where the input is so long and comes from a lot of sources. The length of the input documents makes the RNN-based model ineffective to capture the meaning of the remote words. To deal with this problem, we use two techniques. The first technique, bidirectional network, is so prevalent in NLG problem based on RNN model. Its idea is to apply the RNN model from the start to end of sentence and vice versa. However, with the long documents, its improvement is not extremely significant. Therefore, we propose to combine CNN and RNN for the encoder. The primary goal of this combination is to take advantages of CNN to capture the local features of inputs through convolution and max-pooling operation.

The second problem in Summarization problem based on the learning model is the difference of the measurement in the learning and evaluating systems. In the learning process, the loss function is commonly chosen as the log-likelihood of words in the vocab-

ulary. However, the evaluation measurement is Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [11]. Therefore, we want to apply ROUGE into learning process through Reinforced Mechanism. It is so innovative and novel in Abstractive Summarization. Despite that ROUGE is discrete and is not capable of derivation, Reinforced Mechanism is able to combine it with the traditional loss function.

In the previous works, to improve the quality of encoder, the attention mechanism is often applied to the input's sequence. However, the goal of the systems is the quality of the generated summary. The key question is to improve the fluency and elimination of words in the sequence. Therefore, I apply the Bilinear Attention Mechanism to both input and previous output to improve the current state.

1.3 Contributions

Our main contributions in this paper are:

- Proposal of the automatically abstractive summarization system with the significant performance.
- Combination of two different kinds of neural networks into encoder model to improve the text understanding.
- Integration of bilinear attention mechanism into the input and predicted output to enhance the quality of summary.
- Exploration of the reinforced mechanism for abstractive summarization model which plays an important role in the issues of fluency.

Our thesis is structured as follows. Chapter 2 will describe some background, while our chapter 3 will concentrate on our approaches. Chapter 4 will present our experiments and results, followed by the conclusion in chapter 5.

Chapter 2

Literature Review

2.1 Sequence to Sequence Model

The sequence to Sequence (Seq2Seq) model is the advanced improvement in Deep Learning model in Natural Language Processing (NLP). It is first introduced in Cho et al. [4] and becomes popular in text learning. The key problem in text processing is the length's variation of sentences or documents. With the word representation, Deep Learning model proves the robustness and effectiveness in vector space. However, sentence representation and generation is the huge challenge in NLP.

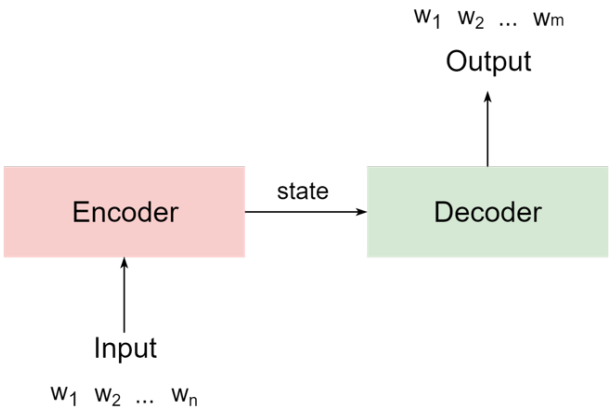

Figure 2.1: The Process of Sequence to Sequence Model

In the technical sense, the Sequence to Sequence model is the combination of two related models which is able to learn the variable length-sized input, which is illustrated in Figure 2.1. Two components are often chosen in Recurrent Neural Networks models and its variants. With the integration of RNN models, Seq2Seq is able to capture the meaning of a sentence or documents into the state which is fed into the decoding phase for sequence generation. With its characteristics, Seq2Seq model is extremely suitable for NLP problem especially in sequence generation task like Statistical Machine Translation

(SMT), Abstractive Summarization, Question Answering (QA) and so on.

The goal of sentence representation is not only to map it into the vector space but also to capture the semantic features in the sequence. With the meaningful representation, the system is able to generate the target sequence in the purpose of the problem. In this sense, the critical issue is to concatenate two phases into a single learning model. It helps the learning process to spread the loss of target goal in two stages and improve them simultaneously.

Another advantage of Seq2Seq model is its flexibility. For both phases: encoder and decoder, we can install them with a lot of models like RNN, CNN, LSTM, and so on. By taking advantages of the different networks, Seq2Seq model is easy to adapt to the variety of domains and areas.

2.2 Recurrent Neural Network

In this section, we will present the overview of Recurrent Neural Network that is so popular in NLP problem. Besides, among a lot of its variants, we focus on Long Short-term Memory network which is the part of our encoder.

2.2.1 Recurrent Neural Network

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks which are inspired by the recursive process. The unique characteristics of RNN are their connections between units constructing a directed graph along a sequence. It allows us to learn the features in the time series of inputs simultaneously.

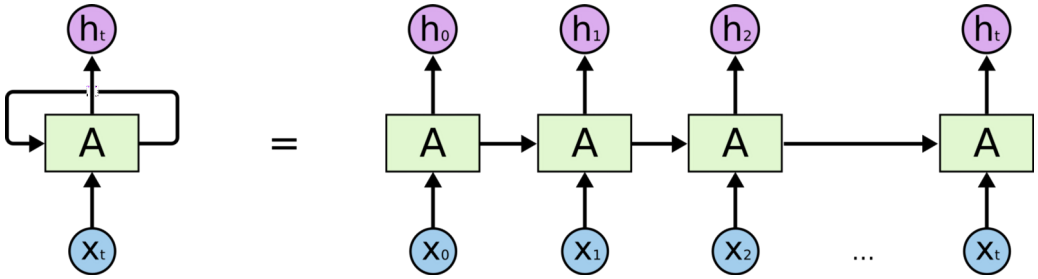

Figure 2.2: The Illustration of Recurrent Neural Network in two forms: rolled (left) and unrolled (right) version. *Source: Towards Data Science Blog*

Like many Deep Learning models, RNN is relatively old. It has initially appeared in the 1980s without any significance. However, in recent year, with the development of computational power, RNN and its variant prove their effectiveness in text processing. Unlike Feed-forward Neural Network (FFNN), it assumes that all inputs (and outputs) are independent of each other. However, in lots of tasks, this assumption is not good

enough for the learning process. To get the relation among the inputs, RNN is the best choice in recent.

Why the model is called the recurrent model. This question reflects the critical characteristic of RNN. In RNN, the computational operation is performed equally to every element in the input. In this operation, the output for each time step is depended on the previous computation, which is presented in the Figure 2.2. It is the reason that it is considered as the memory which captures the information in the previous states. In the theory of RNN, it can make use of arbitrarily long sequences, yet in the practical viewpoint, the length of prior states we want to store is limited in a few time steps.

With their internal memory, RNN is able to remember the feature of the previous inputs to enable it to be precise in predicting the next state. It is the important reason that RNN is so popular in sequential data like time series, speech, text, and even financial data, audio, weather and so on. However, how the model is able to capture the content of previous steps. In Figure 2.2, it is easy to observe that each state h_i is the computational combination of the current input x_i and previous state h_{i-1}. In this sense, the information cycles through a loop. To learn the current representation, it takes into consideration the current input and also what it has learned from the inputs it received previously. Obviously, a Recurrent Neural Network required two different inputs: the present information and the recent past states.

To illustrate its process in computational aspect, we assume that the input sequence is $X = \{x_1, x_2, ..., x_n\}$ and $x_t \in \mathbb{R}^d$ is a input vector of time step t with d-dimension. In the recurrent mechanism, Equation 2.1 is apply n times for each input vector of X.

$$h_t = \sigma\left(W_h h_{t-1} + W_x x_t + b_h\right) \tag{2.1}$$

$$o_t = f(h_t) \tag{2.2}$$

In Equation 2.1, to calculate the state of time step t, RNN requires two inputs: the input vector of time step t x_t and the previous state from the last time step h_{t-1}. If the dimension of the hidden state h_t is $\mathbb{R}^r$, the dimension of the parameter W_h is $\mathbb{R}^{r\times r}$, W_x is $\mathbb{R}^{r\times d}$, and the bias b_h is $\mathbb{R}^r$. To make the learning function flexible, we apply its combination with the non-linear transformation σ like sigmoid function, tanh, ReLU, and so on. Finally, we apply the function f to h_t to get the output for time step t. The parameter W_h and W_x is learned to capture the effect of inputs and previous state through the model's target.

As we present above in both practical and theoretical aspects, RNN can capture the information from the previous states. However, in the Equation 2.1, we can observe that the target state h_t is based on one previous state h_{t-1}, so the traditional Recurrent Neural Network model is considered as the short-term memory network. In practice, RNN meet a critical problem considered as the vanishing gradient [3] which make vanilla RNN can not learn the long-term dependencies. It is the reason that despite its early appearance, RNN

only becomes the interest in recent. The reason for this trend is due to RNN's variants like Long Short-term Memory (LSTM), Gated Recurrent Unit (GRU), and so on.

2.2.2 Long Short-term Memory

To overcome the vanishing gradient descent, Long Short-term Memory (LSTM) proposed by Hochreiter et al. in 1997 [9] is capable of learning long-term dependencies. It works tremendously well on a large variety of task and is so popular in recent in sequence problem. It is considered as the variant of the RNN model, so its mechanism is similar to RNN for repeating the operation in the chains of the input sequence. However, in the RNN model, the operation is effortless with non-linear transformation σ. With LSTM, instead of having a single neural network layer, there are four components for each unit. Their organization gains the huge improvement of LSTM with vanilla RNN.

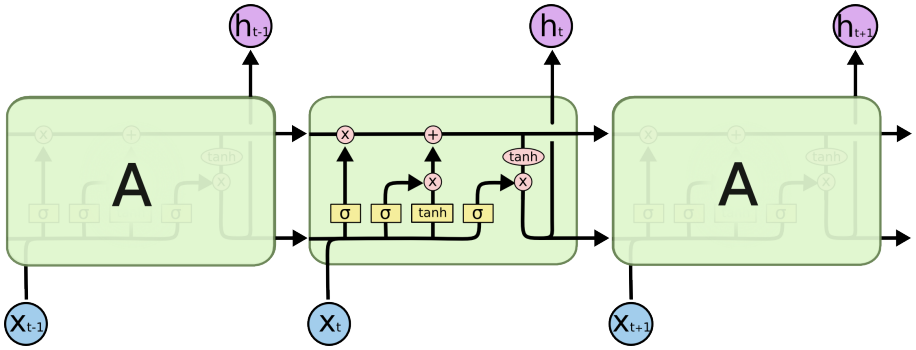

Figure 2.3: The operation of Long Short-term Memory. For each unit of LSTM, there are four non-linear transformation based on the interaction between the input and the previous state. The above flow in the picture is the cell state transmission and the below one is the hidden state. The note of operator and symbols is presented by Figure 2.4 *Source: Colah's Blog*

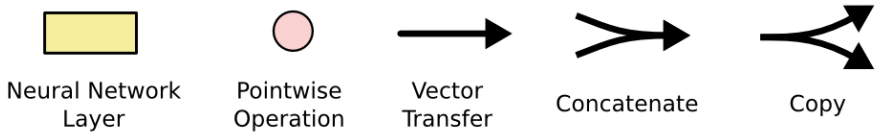

Figure 2.4: The symbols illustration of Long Short-term Memory model *Source: Colah's Blog*

The main idea behind LSTM architecture is the gating mechanism. Through these gates, the model is able to control which information should be kept and transmitted to the next state. Gates are a way to control the information flow in the learning process. In this sense, there are three kinds of the gate in LSTM. It consists of input, forget and output gate. Another difference in LSTM is the cell state. Instead of the hidden state

reflecting the previous information, cell state is the new concept in LSTM which is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions, which is easy for information to flow along it unchanged.

The first step in LSTM is to determine which information is not important and should be thrown away, which is done by forget gate. This decision is made by sigmoid function in the combination of the previous state h_{t-1} and the current input x_t in Equation 2.3. The output of this gate is the probability of information we need to keep.

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{2.3}$$

Where [.] is the concatenating operation in vector space.

The next step is to determine which input's information we want to store in the cell state. Firstly, we calculate the chance to store the candidate information by the input gate controller via Equation 2.4.

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{2.4}$$

Then we define the information we want to learn as the mechanism of RNN model with tanh transformation via Equation 2.5

$$\tilde{C}_t = \tanh\left(W_C\left[h_{t-1}, x_t\right] + b_C\right) \tag{2.5}$$

Accordingly, the new value of state is the combination of candidate information and the remaining information from the previous state via Equation 2.6

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

Finally, we have to calculate the output for each cell and the hidden state for spreading to the next computation. We note that this output is just the internal cell which is not similar to the output of one step. This output is the product of probability function (Equation 2.7) and the non-linear transformation of cell state information (Equation 2.6).

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{2.7}$$

$$h_t = o_t * \tanh(C_t) \tag{2.8}$$

Through these gating mechanisms, gradients related to memory C_t is maintained for a long time, which based on the learning parameter W_t, W_i and the non-linear transformation. The key idea in this mechanism is the selective learning and remaining. Through the learning process, LSTM can make a decision on selecting the important information which needs keeping. In this sense, it can solve the problem of vanishing gradient in vanilla RNN. Although there are a lot of variants of the LSTM model, we only present it briefly instead of all variants which are unrelated to my proposed model in Chapter 3.

2.2.3 Bidirectional Long Short-term Memory

Although LSTM is better than vanilla RNN, it is still less effective in the long sequences. Especially, in Natural Language Processing, not only is the length of the sequence so long but also the relation of internal elements is extremely complicated. In the traditional LSTM, for each time step, the hidden state is depended on the previous state on the left as Equation 2.7 and 2.6. It means that the features of the current state are just based on its input information and its relation with the previous states. However, in the sense of NLP, the meaning of one state such as word and sentence is depended on its content which is the surrounding states on both the left and right sides. Therefore, to meet this requirement, the bidirectional RNN model proposed by Schuster et al. [21] is considered as the inspiration of bidirectional trend in sequence learning.

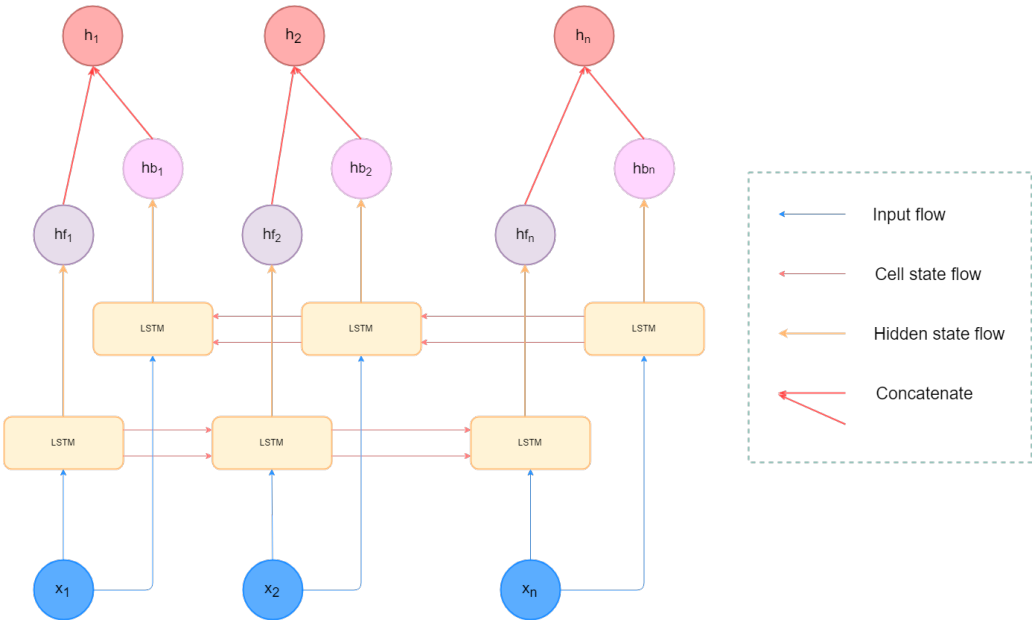

Figure 2.5: The bidirectional Long Short-term Memory model. In this model, the node LSTM is the representation of one unit which is specifically presented in Section 2.2.2. In common sense, the final hidden state of BiLSTM is just the concatenation of forwarding and backward ones. The direction of cell state flow reflects the side of the previous information in the learning process. In this figure, the output layer is eliminated for the simplicity's illustration

To take advantages of Bidirectional techniques and LSTM, Bidirectional LSTM (BiLSTM) is proposed to overcome the challenge in sequence learning. The detail of BiLSTM is presented in Figure 2.5. In the common sense, it is the combination of two LSTM layer with the concatenating operator. The computational operation is presented in Equation 2.9, 2.10, 2.11.

$$h_t = [h_t^f, h_t^b] \tag{2.9}$$

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

$$h_t^b = LSTM(h_{t-1}^b, x_t) \tag{2.11}$$

BiLSTM model takes advantages of the strength of LSTM in bidirectional learning. It is extremely powerful in text understanding where the representation of words is tightly dependent on their content. It means that BiLSTM model is able to get the global features of words in the creation of sentence representation.

2.3 Convolution Neural Network

Unlike Recurrent Neural Network which is so popular in text processing, the network I want to present in this Section is often applied into Image Processing. Inspired by the visual cortex of mammals which help them perceive the world in their brain, Convolution Neural Network (CNN or ConvNet) is proposed in the 1980s as "neocognitron". They are also known as shift invariant or space invariant Artificial Neural Networks (SIANN), which is based on their shared-weights architecture and translation invariance characteristics.

ConvNet is the hierarchical neural network with one or more convolutional layers followed by one or more fully connected layers as FFNN. With the convolution operation, the information is twisted and integrated into the new state. The key question is how the convolution operation works. In the mathematical sense, convolution is the mapping function f which is defined by Equation 2.12.
Given f, g: $\mathbb{R}^2 \to \mathbb{R}$ and the set $A = \{(a, b)|g(a, b) \neq 0\}$

$$(g * f)(x, y) = (f * g)(x, y) = \sum_{(a,b)\in A} g(a, b)f(x - a, y - b) \tag{2.12}$$

This operation is the integral measuring how much two functions overlap as one passes over the other. In the simple sense, two functions are considered as being "rolled or twisted together". Thanks to this characteristic, convolution operation is often used for image smoothness, edge detection, and so on. It is one of the reasons we consider CNN as space invariant.

The strength of ConvNet is to take advantages of the geometric relation. Its assumption is that the points in the local area have the semantic association. The role of convolution operation is to catch the local features, emphasize them, and eliminate the unnecessary points, which is extremely suitable for Image Processing.

Specifically, ConvNets consists of two components: the feature extraction and classification. In this section, we just focus on the first component - feature extraction. To extract the feature, ConvNets have three main definitions: Local Receptive Field, Shared

Weight, and Pooling.

Local Receptive Fields (LRFs) are the special definition of ConvNet. It is defined as the fix-sized region in the input space where the convolution operation is applied. The size and duplication of these regions are based on the step and filter in ConvNet. It means the input space is partitioned into the different parts where the information is integrated to form the hidden state in the next step of learning as Figure 2.6.

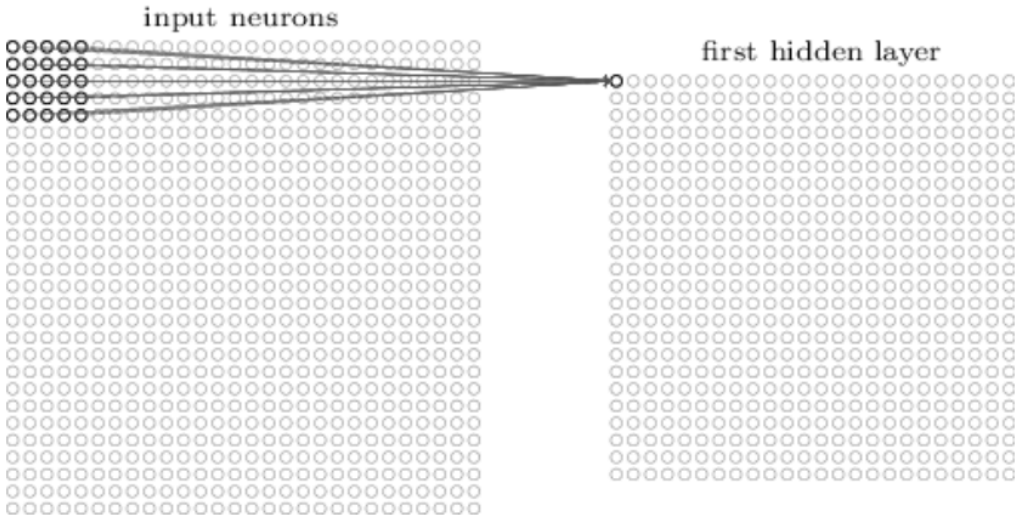

Figure 2.6: ConvNet: Local Receptive Fields: The left matrix is the input space with the size of 28x28. The Local Receptive Fields is the bold black region with the size of 5x5. This region is transformed into the respective state of the hidden layer in the right matrix

Another component in ConvNet is shared weights. As we mentioned above, the convolution operation is the key mapping function in ConvNet which requires two elements. The first one is the input and the second is called the filter or kernel. For each Local Receptive Field, the filter is applied to get the expected features in this region through the convolution transformation. The value of filter is called the weights or the parameters of ConvNet. Unlike the other neural network, in ConvNet, the weights of each filter is used for all LRFs. It is the reason we consider it as shared weights. This technique helps us to discover the non-observable features in the input space. Another advantage is the save for the computational cost. Instead of the huge parameters in the other neural network, in ConvNet, the model just needs to find the weight for each filter whose size is the same as LRFs. It is extremely significant in practice especially for the huge volume of data in text and image processing.

The main reason that ConvNet is considered as the shift invariant is the effect of pooling operator. Pooling is the family of functions used to aggregate the information in the specific regions. In image processing, the small changes like pixels' difference are often useless to recognize. Therefore, pooling in the small area is the way to emphasize the hidden features and maintain them to spread to the next layer. Besides, pooling

operator is to progressively scale the spatial size of the next layer's representation which means that the number of parameters and computational cost are reduced significantly. It also helps us to control overfitting phenomena in learning problem. There is a variety of pooling operation such as Max, Mean pooling and so on. For example, in Max Pooling, the new state is created by the max value in the region as the Figure 2.7

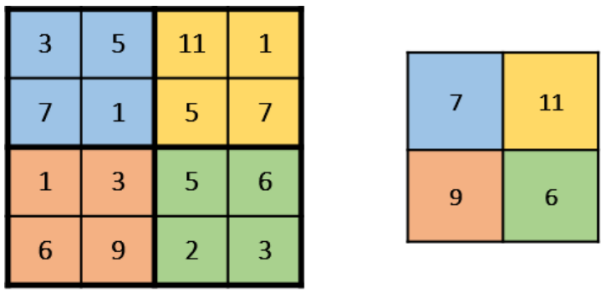

Figure 2.7: ConvNet: Max Pooling Example: In two-sided figure, each local region is colored by the different colors. The max pooling is applied in the left one to create the right one

In the above, we talk some techniques in ConvNet through the sense of Image Processing. In this part, we focus on ConvNet in Natural Language Processing. The main difference between the two areas is that the relationship of components in the input space. In NLP, the input is the text sequence whose elements are words. In common, those words are mapped into the vector space by Word2Vec [14] or Glove [19]. In this sense, the input space is the combination of word representations in the sequence like Figure 2.8

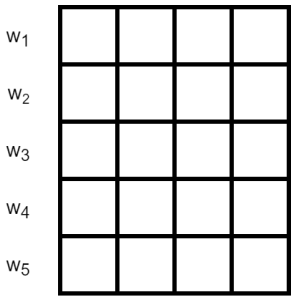

Figure 2.8: The input space of NLP task: For example, with each word w_i, we present it in the 4-dimension vector and the length of sequence is 5

In the term of visuals, the representation of the sequence in Figure 2.8 is similar to the image. In the image, the pixels are independently meaningful. It means that each pixel contains its content. However, in the sequence, the value of elements in the same row is dependent and has no ability to consider independently like the pixels. Therefore, in NLP task, we only use 1-dimension convolution operator. It means that the kernel or filter is chosen by the size of the multiplication between the dimension of the word vector and the number of words in Figure 2.9

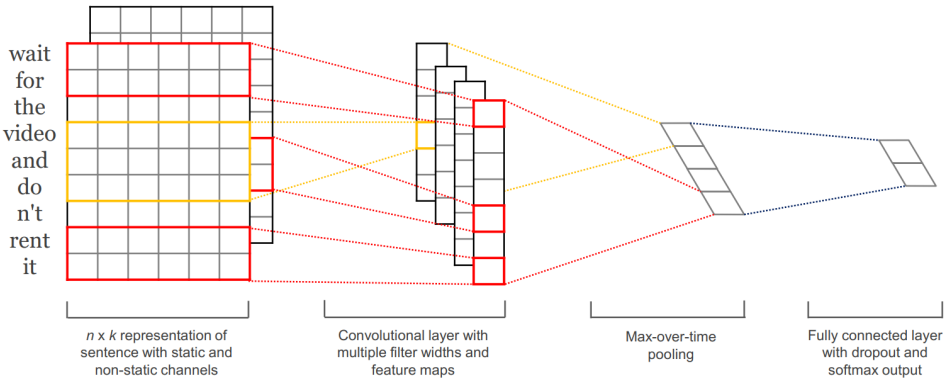

Figure 2.9: The example of ConvNet for sentence classfication task: This is the model proposed by Yoon Kim [10]. In this model, 1-d convolution is applied into the word representation matrix to get the local features in the sentence.

In NLP, the sentence is often considered in n-grams which is similar to the filter's mechanism in ConvNet. It proves that ConvNet is capable of applying in the NLP task, especially for sentence representation. The main advantage of ConvNet in NLP is its ability to explore the local features in the sequence. Its local features are the combination of word's meaning and its content with surrounding information.

2.4 Attention Mechanism

In the text processing, the huge problem is the length of the sequence. If the sequence is too long, we do not know which words or phrases are important, even in the human processing. The previous model we mention above just treats equally among the components. It means all words in the sentence are crucial or rather, no word is important. In fact, this treatment is not good for the long sequence where it always exists the main words or phrases affecting the meaning of sequence. Especially, in Summarization task, the goal of the expected solution is to find the key sentences or phrases in the documents for the summary.

To overcome this problem, Attention Mechanism is used as the important score of each element. In this sense, the components which have a lot of contribution to the sentence representation is a higher attention-score and vice versa. In practice, attention is simply a vector whose outputs often is the result of the dense layer using softmax function. For this purpose of introduction of Attention Mechanism, we focus on Neural Machine Translation (NMT) model as the specific example.

Machine Translation (MT) task is also the sequence to sequence problem. The goal of this task is to translate the source sentence into the target one in the different language. Without Attention Mechanism, the translator treats all words equally, which is not suit-

able in practice. For translating, attention allows machine translator to look over all the information to zoom in or out the features which affect the meaning of the target sentence.

In the NMT model proposed by Bahdanau, 2014 [1], they use LSTM-based Sequence to Sequence model in two phases: encoder and decoder. They use the final hidden state in the encoding phase to generate the new sequence step by step. The detail of their system without attention is presented in Figure 2.10

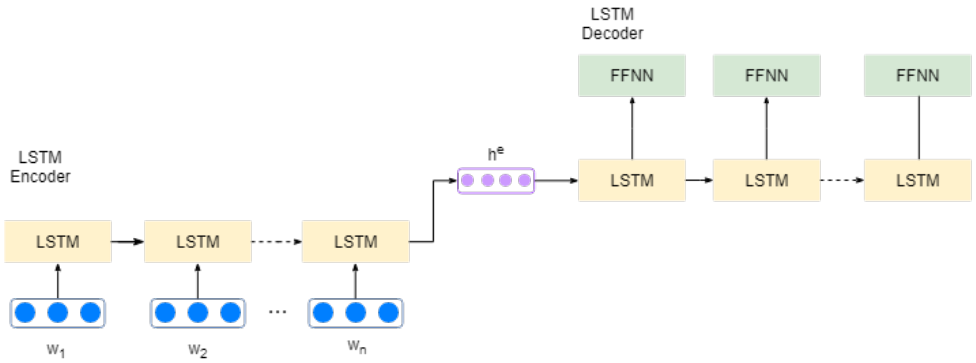

Figure 2.10: LSTM-based Sequence to Sequence Model for Neural Machine Translation: This figure is inspired from the Bahdanau's work [1]. In this model, the input is integrated gradually into the final hidden state h^e which is fed into the LSTM model in decoder

In the above model, the importance of all words is equal to create the representation of h^e. In Attention Mechanism, for each hidden state of LSTM encoder, the importance's score is calculated by Equation 2.13 and s_i (Equation 2.14) is the hidden state of LSTM decoder in time step i.

$$\alpha_{ij} = softmax(e_{ij}) = \frac{\exp e_{ij}}{\sum_{k=1}^{T_x} \exp e_{ij}} \tag{2.13}$$

where

$$e_{ij} = f(s_{i-1}, h_j) \tag{2.14}$$

After that, the content of input sequence is defined by the sum of all hidden state and its weight in the LSTM encoder like Equation 2.15. This context vector is fed into LSTM model to predict the word in the output in each time step i.

$$h^e = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.15}$$

In short, Attention Mechanism is the way to determine the weight of components. These importance scores reflect the contribution of these elements into the context vector which is used for the next step. There are a lot of attention techniques based on the scoring function f where two favorite kinds of the mechanism are additive attention [1] and multiplicative one [12]. In our proposed model, we use the bilinear multiplicative attention which is presented specifically in Chapter 3.

2.5 Reinforcement Learning

In recent years, Reinforcement Learning is considered as the new and promising technique to get the computer closer to a human being. In this sense, the computer can interact with the impact of the real world. Especially, the success of AlphaGo system [23] in the real competition is the milestone in the trend of Machine Learning.

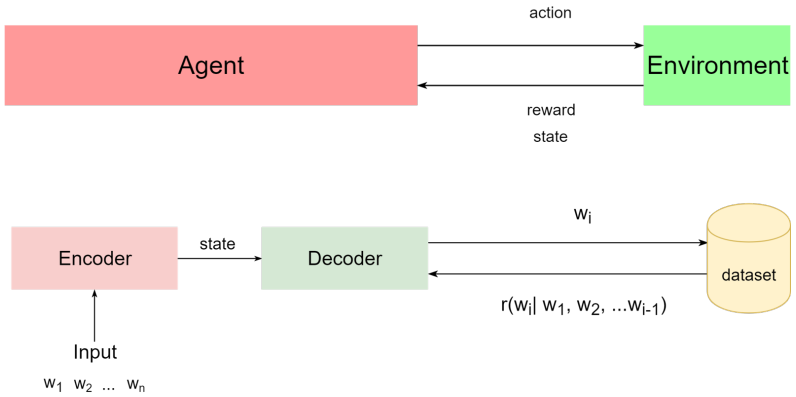

Figure 2.11: The correspondence between Reinforcement Learning and Sequence to Sequence model

One of the main question in my research is how to integrate Reinforcement Learning in Abstractive Summarization problem. As the Figure 2.11, the response of Reinforcement Learning to the environment and the learning process of Sequence to Sequence model is quite similar. If we consider the generation as the action of the agent and the evaluating score of output as the reward, the integration capability is absolutely possible. However, the key question in learning process is how to combine the reward function into the back-propagation. The difficulty is that the loss function is smooth and derivable, yet the reward function is often discrete and irreducible.

In this part, we focus on the self-critical policy gradient training algorithm [20]. It is one of the techniques to integrate the reward function into the gradient descent training. In this technique, the reward score is multiplied with the original gradient in Equation 2.16. The idea of this technique is to add the difference between the expected output $\hat{w}$ and the sample output w^s from your system. The detail is presented in Figure 2.12.

$$\frac{\partial L(\theta)}{\partial s_t} = (r(w^s) - r(\hat{w}))\,(p_\theta(w_t|h_t) - 1_{w^s}) \tag{2.16}$$

Where s_t is the input to the softmax function and 1_{w^s} is the 1-vector with the same dimension of w^s.

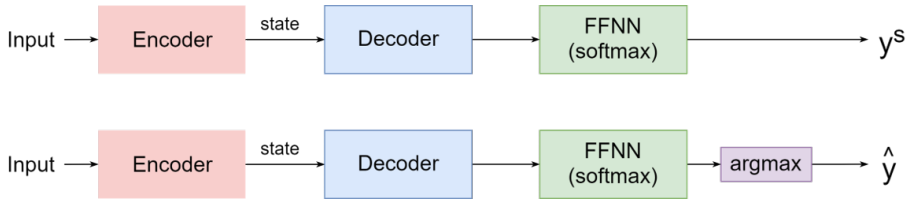

Figure 2.12: The self-critical mechanism in Reinforcement Learning: The visualization in self-critical learning where w^s is the direct result of softmax layer for the prediction in a time step t and $\hat{y}$ is the result of argmax operator

2.6 Related Works in Abstractive Summarization

With the purpose to prove the robustness of our proposed model, this section presents the previous works that related to our study. Firstly, we want to talk about our inspiration for my search. It is the work of Romain Paulus et al. [18]. The architecture of their model is shown in Figure 2.13. In the quick observation, this model and our model are quite similar. However, the main difference between the two models is the encoder model. In their work, they only use BiLSTM to embed the document while we combine two kinds of the network for encoding phase. It is considered the first work of Reinforced Mechanism in Abstractive Summarization. With their pioneering, we develop the reward function for this problem as the combination of different score that is more effective to control the quality of the generation.

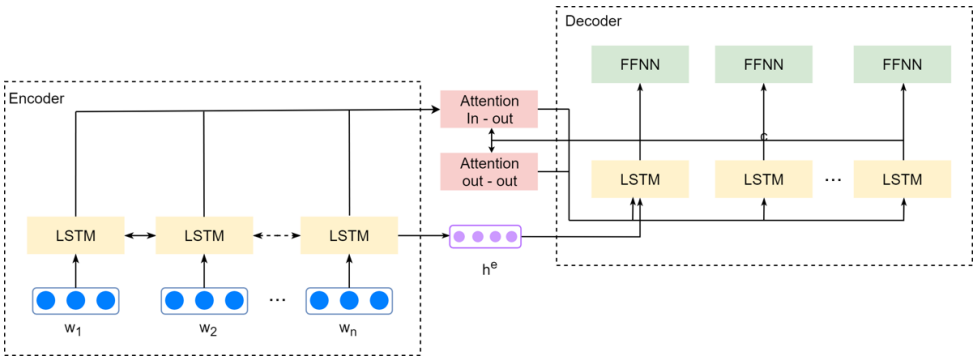

Figure 2.13: A Deep Reinforced Model of Romain Paulus et al. [18]: In their model, they use BiLSTM and LSTM for encoding and decoding phase. They apply Bilinear Attention and Reinforced Mechanism into the generating phase.

The next system we want to discuss is the SummaRuNNer [16] system whose special advantages is the new problem representation. For Extractive Summarization task, they consider it as the sentence binary classification problem. The structure of their model is shown in Figure 2.14. In this model, they use the bidirectional Gated-Recurrent Unit model to encode the sentences. The document vector is created by the sum of all final hidden states in the sentence encoding phase with the *tanh* activation. The combination

of document information, current sentence state, and the previous decision are fed into the binary classifier to decide whether this sentence should be chosen or not. The strength of this work is the formation of document representation. After the learning process, they can produce the document vector that is very necessary for some other tasks. The second advantage of this model is its flexibility. Although this model is designed for Extractive Summarization, it is able to change for Abstractive Summarization. In SummaRuNNer-abs, with the document representation in the previous step, they use it to feed into the RNN-decoder model to generate the abstractive summary. In their sense, they also use GRU model for decoding phase. In the experiment, they propose to use the Lead-3 model which simply produces the summary from top-3 sentences of the document based on the probability of decision in the classifier.

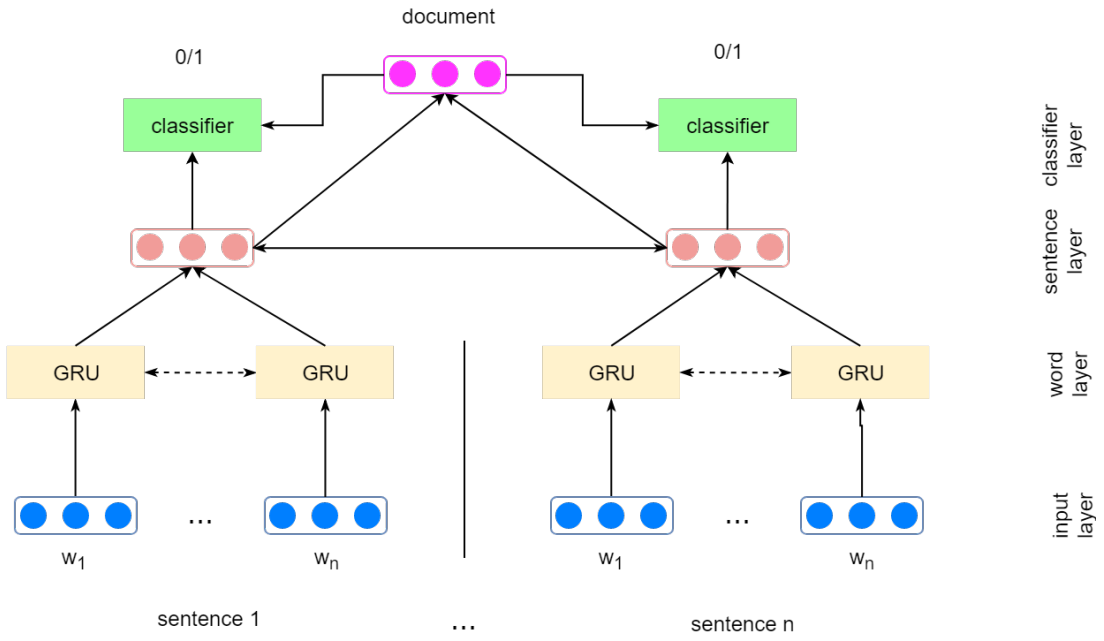

Figure 2.14: The architecture of SummaRuNNer in Extractive Summarization task: In the word level layer, the sentence is encoded by bidirection GRU model. After, this representation combined with the document one is fed into the classifier to make a decision on choosing for the summary.

The final system we want to talk in this section is Pointer-Generator Network whose ROUGE-2 is significantly high. The structure of this model is given in detail by Figure 2.15. In this model, they use the traditional encoder-decoder model by BiLSTM-LSTM network for the main architecture. Their novelty is the integration of the pointer mechanism in the decoding phase. In each time step, the probability of generation p_{gen} is calculated by the context vector of the encoder, the previous decoder states, and the current one. The new word in the summary is decided from the copying from the input and generating from the vocabulary options by p_{gen}. It helps the generator to solve the problem of the vocabulary's limitation by copying the word in the input.

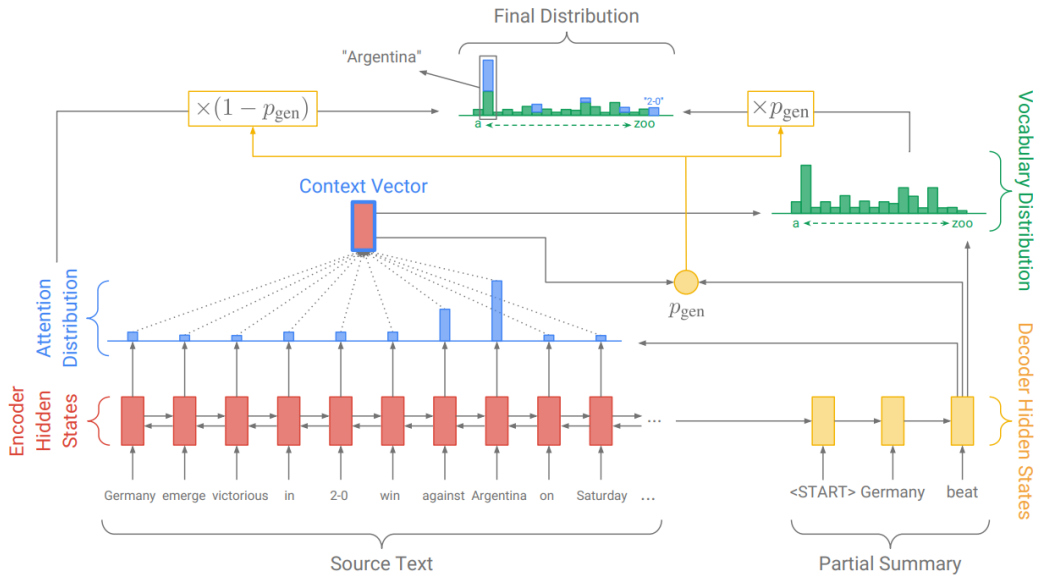

Figure 2.15: In this model, they use the BiLSTM-LSTM model for encoding and decoding phase. For attention mechanism, they apply the multiplicative one in the input to get the important score of these words. In the decoding phase, the new word is formed by the decision either copying the input or generating through pointer mechanism.

Chapter 3

Our approaches

In this chapter, we focus on the detail of our proposed models in our study. Besides, some techniques in the implementation's sense are also presented to make clear the operating process of our model.

3.1 Encoder model

Among many previous approaches in Abstractive Summarization, we choose Sequence to Sequence model as the main flow of our model. The key advantages of its model are its robustness and scalability. In this sense, we can integrate any models for the sequence understanding and generating phases. Besides, with the strength of Deep Learning model, the model does no requirement of hand-crafted features which is the barrier to put it into practice. In this section, we present the encoder model that is the combination of LSTM and ConvNet to embed the sentence.

In the early of our study, our encoder model only consists of Long Short-term Memory with bidirectional technique to capture the meaning of the sentence. In this model, we consider the input document as the sequence of words. The role of the sentence in the document is eliminated by the sequence representation. After, we use the word embedding matrix to map each word into the vector space. The final hidden state of our encoding phase is the concatenating of forward and backward learning as Equation 2.9. All components of this process are illustrated in Figure 3.1.

However, as we mention in Chapter 2, BiLSTM is just able to capture the global features of the sequence. Especially, in Summarization task, the length of documents is so long that the meaning of words and phrases in the remote position in the sequence is significantly blurred. To overcome this problem, we propose to take advantages of ConvNet to capture the unique local features of the n-gram in the sequence. However, in text understanding, each composition of words has the different content. Hence we propose to use multi-channel of ConvNet in the encoding phase. The reason for this technique is that each channel is corresponding to the n-gram phrase which represents

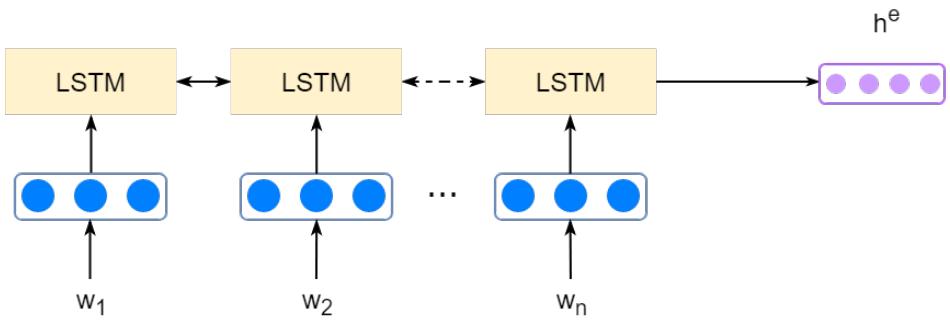

Figure 3.1: BiLSTM model for encoding phase: Firstly, each word is vectorized by the word embedding matrix. After, these vectors are fed into LSTM unit in bi-direction to get the final hidden state h^e as Equation 2.9

the important and necessary regions in the document. In this sense, we combine two kinds of networks: BiLSTM and Multi-channel ConvNet to get both local and global features in the sequence which is presented in Figure 3.2.

In multi-channel ConvNet, we also use the 1-d convolution operator sliding through the sequence from left to right. After the filtering, the result of convolution and max-pooling operator is flattened and fed into the Feed-forward Neural Network. The purpose of FFNN is to scale the feature vector by the dimension of h^{bilstm} and make it flexible by non-linear transformation. The final hidden state of encoding phase is the combination of h^{bilstm} and h^{mcnn} as Equation 3.1.

$$h^e = h^{bilstm} \oplus h^{mcnn} \tag{3.1}$$

Where $\oplus$ is the vector addition operator.

The novelty of our encoder model is the combination of two different networks: BiLSTM and ConvNet. It helps us to capture both global and local features in the sequence. Therefore, the representation of the sequence is more meaningful and flexible. Another advantage of this expansion is the computational cost. As we mentioned above, the time processing of ConvNet is faster than LSTM-based networks, hence we only pay a small extra cost in exchange for the apparent advantages of ConvNet.

3.2 Decoder model

After encoding the documents, the context vector of the sequence needs feeding into the decoder model to generate the expected sequence as the summary. Unlike the encoding phase where we combine two different networks, in the decoding phase, CNN-based models are not suitable, so we focus on the RNN-based models. Specifically, we use LSTM model and Feed-forward Neural Network with softmax layer to generate the words in the time series. The initial state of LSTM for decoder model is the hidden state of encoding phase h^e. Based on the previous state and the context of the input sequence, the de-

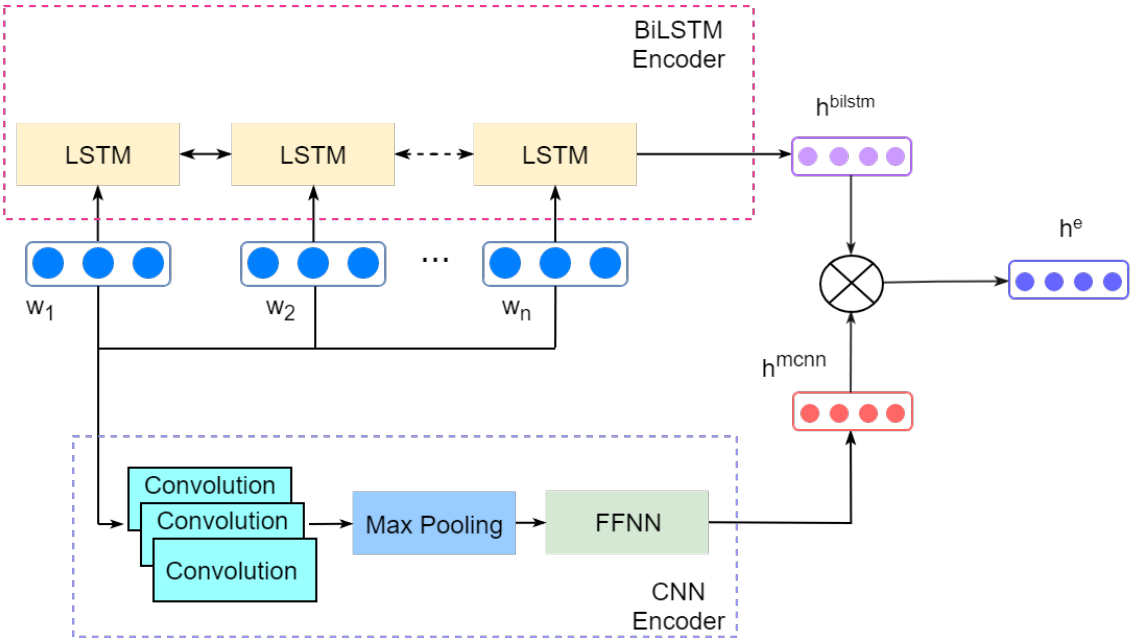

Figure 3.2: The illustration of encoder model: The sequence of word vector is fed into two models: BiLSTM and Multi-channel ConvNet to embed them into h^{bilstm} and h^{mcnn}. The final hidden state of encoding phase is the combination of two hidden states via Equation 3.1

coder works step by step to generate the hidden state h_t^d. These states are fed into the Feed-forward Neural Network to determine the probability distribution of words in the vocabulary. The detail of this model is presented in Figure 3.3.

In the computational sense, the output in each time step t is determined by softmax layer as Equation 3.2, 3.3.

$$a_t = W_V h_t^d + b_V \tag{3.2}$$

$$y_t^j = softmax(a_t)_j = \frac{e^{a_t^j}}{\sum_{k=1}^{V} e^{a_t^j}} \tag{3.3}$$

3.3 Bilinear Attention Mechanism

As we mentioned above, the length of the sequence is the key problem of text understanding. As the sentence is too long, the traditional networks are useless to capture all the information into the limited space. Despite our combination in the encoding phase, it seems no enough in Abstractive Summarization. The inside reason is that most of our concern is the encoding phase. We make an effort to represent the input sequence into the meaningful space. Therefore, to overcome this problem, we apply the Attention

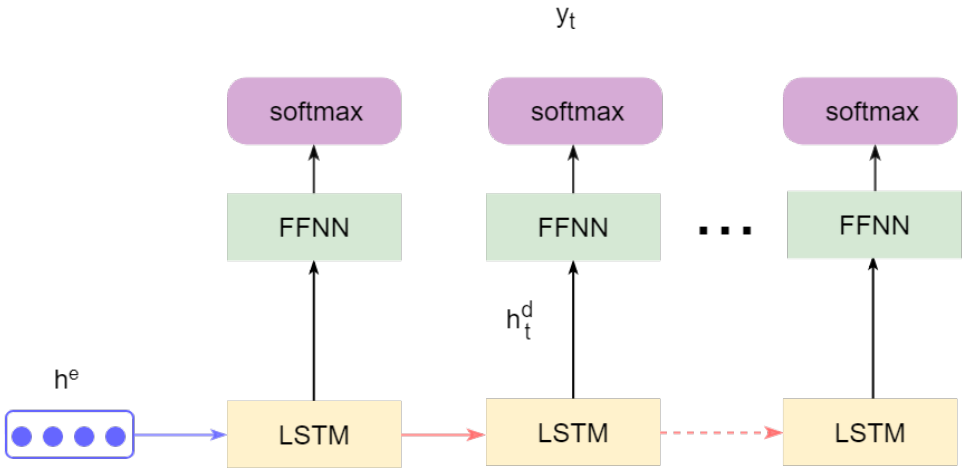

Figure 3.3: The illustration of decoder model: The encoder state h^e is fed into the LSTM model as the initial state. Each decoding state h_t^d is applied by FFNN and softmax layer to the output y_t as the word probability distribution in the vocabulary.

Mechanism into our proposed model.

In our model, instead of additive attention, we use bilinear multiplicative one. In this technique, the scoring function of attention is based on the interaction of two components, and the linear function is the multiplication. In our model, we use two different kinds of bilinear attention. The first one is the intra-temporal attention on the input sequence. For each time step, the generator considers the contribution of each word in the input sequence and the hidden state to make a decision on the word prediction. With the long source sequence, this attention helps us to focus on the important local regions in the input instead of the equal treatment in the previous version, which reduces the redundancy in the document. The mathematical viewpoint of this attention is presented in Equation 3.4, 3.5, 3.6. Finally, the context vector of the input sequence is the sum of all multiplication of the hidden state and the corresponding attention score as Equation 3.7.

$$e_{ti} = f(h_t^d, h_i^e) = h_t^{dT} W_{attn}^e h_i^e \tag{3.4}$$

$$e_{ti}^{'} = \begin{cases} exp(e_{ti}) & \text{if t = 1} \\ \frac{exp(e_{ti})}{\sum_{j=1}^{t-1} exp(e_{ji})} & \text{otherwise} \end{cases} \tag{3.5}$$

$$\alpha_{ti}^e = \frac{e_{ti}^{'}}{\sum_{j=1}^{n} exp(e_{tj}^{'})} \tag{3.6}$$

$$c_t^e = \sum_{i=1}^{n} \alpha_{ti}^e h_i^e \tag{3.7}$$

Obviously, we also put our concern on the effect of input sequence on the output prediction. However, the internal relation of output has not been exploited yet. It means

that the current predicting decision depends on the previous states in the output. Especially, the problem of repetition in the summary which leads to the weakness of language's fluency is prevalent and difficult in Abstractive Summarization. In the second attention score, we focus on the interaction between the predicting decision and its history in the decoding process. It is called as intra-decoder attention.

This attention is the result of bilinear multiplication of the current state h_t^d and the previous ones h_j^d. Firstly, we define the scoring function between two hidden states as Equation 3.8 which is normalized by Equation 3.9. Finally, the context vector of the previous decoding states is calculated by Equation 3.10. By considering the effect of the previous decoding output, the generator can avoid the repetition in the prediction, hence our summary is less redundant and more fluent.

$$e_{tt'}^d = h_t^{dT} W_{attn}^e h_{t'}^d \tag{3.8}$$

$$\alpha_{tt'}^d = \frac{e_{tt'}^d}{\sum_{j=1}^{t-1} exp(e_{tj}^d)} \tag{3.9}$$

$$c_t^d = \sum_{j=1}^{t-1} \alpha_{tj}^d h_j^d \tag{3.10}$$

The next question in our research is how to integrate these attention mechanisms into our model. For a solution, we concatenate two context vectors which reflect the effect of input and previous outputs with the current predicting decision and put them into the generator layer - softmax layer. It means the probability of words in the vocabulary is defined by the softmax of three components: the hidden states of the decoder model and two context vectors from the attention mechanism as Equation 3.11.

$$p(y_t) = softmax(W_V[h_t^d; c_t^e; c_t^d] + b_V) \tag{3.11}$$

Where [.] is the vector concatenation operator.

In short, my proposed model is the combination of three different components: encoder, decoder, and attention mechanism. With the powerful encoding representation, the decoder generates the words in the sequence with the constraint of two attention scoring function. The structure of our final model is illustrated in Figure 3.4. Although we combine many components in our model, each one is the solution of the different problem in Abstractive Summarization. Besides, the strength of our model is the reasonable complexity. Instead of increasing the size of LSTM model, we narrow and compensate it with the lighter load model – ConvNet. It makes our model more compact and easier to put it into practice.

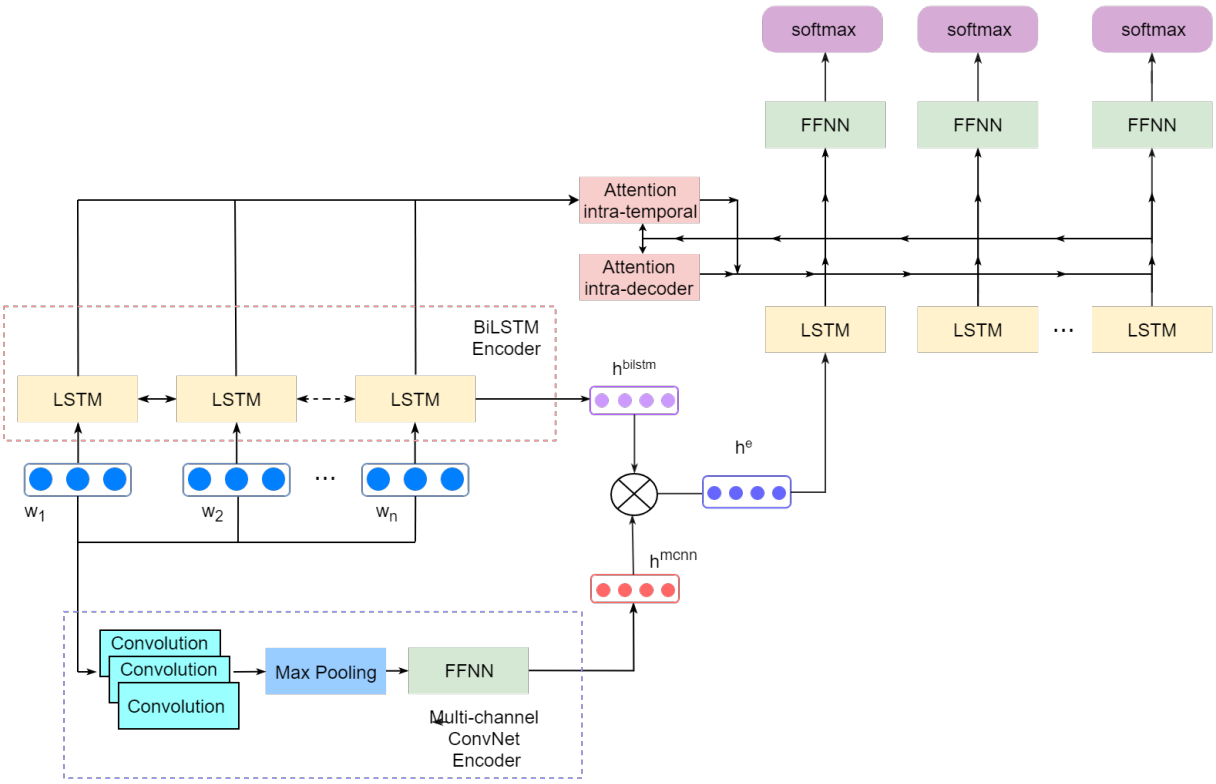

Figure 3.4: Our proposed model with attention mechanism: Our complete model is the combination of three components. In the left-hand side, the encoder model embeds the input sequence into the hidden state h^e which are fed into the decoder as the initial states. The context vectors and the hidden state in decoding phase are put into FFNN to get the probability distribution of words in the vocabulary in each time step

3.4 Reinforced Mechanism

The last and the most crucial technique we use in our proposed model is Reinforce Mechanism. As we mentioned above, in the success of AlphaGo [23], Reinforcement Learning becomes the new trend in Machine Learning. Especially, with the problem where the loss function and evaluation's measurement is different, Reinforced Mechanism is extremely reasonable.

In Abstractive Summarization, we use Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [11] for evaluation. However, in the learning process, our model is trained to minimize the loss function based on the conditional log-likelihood of output y_t. This work comes from the assumption that the more accurately we predict, the more ROUGE-score we get. Apparently, our expected and trained systems are flowing in two different directions. To get two systems closer and closer, we apply Reinforced Mechanism into our system. Our goal is to integrate ROUGE score into loss function. Among a lot of techniques, we use self-critical policy gradient training which is presented in Chapter 2.

Firstly, we define the reward function of the predicted sequence by ROUGE score. In summarization, the quality of the system is often evaluated by three different ROUGE scores which include ROUGE-1, ROUGE-2, and ROUGE-L. The characteristics of ROUGE-1 and ROUGE-2 is the same as conditional log-likelihood loss function because they consider the overlap of unigram and bigram between the predicted and target words, which ensures the correctness of the generation in each time step. In the other hands, ROUGE-L focus on the length of common subsequence between two sequences. Obviously, this score measures the semantics of the predicted sentence. If the generation is effective, the length of subsequence overlapping with the target is large. To take advantages of these scores, we combine them into the reward function as Equation 3.12.

$$r(s,t) = \frac{ROUGE-1(s,t) + ROUGE-2(s,t) + ROUGE-L(s,t)}{3} \tag{3.12}$$

To optimize our model, we combine two different kinds of the loss function. Firstly, as the previous works in sequence generation problem, we minimize the maximum-likelihood loss in each decoding step. The loss of one-time step is the negative probability of the output in Equation 3.13. This function makes our model more accurate in prediction for each word in the sequence.

$$L_{ml} = -\sum_{t=1}^{m} \log\left(y_t | y_1, y_2, ..., y_n, x\right) \tag{3.13}$$

However, as I mentioned above, this loss function L_{ml} is often not able to produce the best result corresponding to the discrete measurement like ROUGE score. The remedy in our model is to use a self-critical policy gradient learning algorithm [20]. In the training process, we create two output sequences instead of only one in the previous system. The first output y^s is also the probability distribution of words in the vocabulary which is the main result we want to learn. The second one $\hat{y}$ is produced by maximizing the probability distribution of words in the first one. For each output, we take the reward score with the target y^* via Equation 3.12. Finally, we integrate these reward scores into the loss function for training in Equation 3.14.

$$L_{rl} = (r(\hat{y}, y^*) - r(y^s, y^*)) \sum_{t=1}^{m} \log\left(y_t^s | y_1^s, y_2^s, ..., y_n^s, x\right) \tag{3.14}$$

To integrate the strength of two functions, we define the final loss function for training through the linear combination of two components in Equation 3.15.

$$L_{mixed} = \gamma L_{rl} + (1-\gamma) L_{ml} \tag{3.15}$$

Where γ is the trade-off hyper-parameter of model.

The drawback of the reinforced loss function is that it does not consider the readability and fluency of the prediction. It means that ROUGE is just based on the separate n-gram and phrase and not concerns the previous steps. In the other hands, the traditional

objective function L_{ml} works in the same way as the language modeling based on the states of the sequence. In this sense, the order of words in the generating step contributes to the learning objective. Therefore, we combine it into the final loss function to increase the balance of two goals consisting of precision and fluency in the summary.

Chapter 4

Evaluation

4.1 Dataset and Preprocessing

The dataset we used in this study is CNN/Daily Mail[1] [8]. In the previous dataset like DUC [2] and Gigaword [3] whose summary is so short with one sentence for each sample. With the purpose to prove the robustness of our approach in the long-text understanding, we choose CNN/Daily Mail as the main dataset for all experimental models.

In the first appearance, CNN/Daily Mail dataset is used for the passage-based Question Answering task [8]. In the original form, this dataset is collected from the articles in CNN and Daily Mail. Each article consists of the long document and some corresponding highlights which are considered as the main idea of the above passage. Based on these documents, they eliminate some entities and generate the questions whose answer's type is the fill-in-the-blank. For Abstractive Summarization, we use the modified version based on the preprocessing of Nallapati et al. [17]. In this sense, they consider the highlights as the multi-sentence summary of the input document. The detail of the dataset is presented in Table 4.1.

	Train		Validation		Test	
	Doc	**Sum**	**Doc**	**Sum**	**Doc**	**Sum**
Number of samples	287226		13367		11489	
No. Word/sample	792	55	770	62	779	58

Table 4.1: The detail of CNN/Daily Mail dataset.

As we can see in Table 4.1, the number of the document is so large and various with about three hundred thousand samples for training and fifteen thousand ones for testing and validation. It means if the model is too complex, the training time is too long to put into practice. Besides, the number of token per document in this dataset is around eight

[1] https://github.com/abisee/cnn-dailymail
[2] https://duc.nist.gov/data.html
[3] https://github.com/facebookarchive/NAMAS

hundred. Its length is the huge problem for text understanding in sequence encoding phase. Another problem for Abstractive Summarization in this dataset comes from the length of the target summary, which requires our decoder to ensure the correctness and fluency in the generating sequence. These above characteristics are the reason that we choose it as the main dataset in our experiments.

4.2 Experiment Settings

To prove the effectiveness of our proposed model, we do the experiments on the following models:

- RConvBiLSTM (Reinforced Convolution - Bidirectional Long Short-term Memory): In this model, the encoder model is the combination of ConvNet and LSTM like the Figure 3.2. In the decoding phase, we use intra-temporal and intra-decoder attention in Chapter 3.3). For the training process, the loss function is based on the Reinforcement Learning via Equation 3.14.

- ConvBiLSTM (Convolution - Bidirectional Long Short-term Memory): This model is similar to RConvBiLSTM in the encoder and decoder components. The difference between the two systems is the loss function for training. In this model, we only use the function based on the conditional log-likelihood of sequence in Equation 3.13.

- COConvBiLSTM (Combined Objective for Convolution - Bidirectional Long Short-term Memory): This model is applied by all the proposed mechanism in my study. The learning model is similar to Figure 3.4. The loss function in this model is the combination of reinforced and conditional log-likelihood mechanism like Equation 3.15.

The reason for this division is to focus on our effort to prove the effectiveness and robustness of our proposed mechanisms and models for Abstractive Summarization. With this experiment's design, we can evaluate the effect of Reinforced Mechanism into the learning process. In the other hands, the strength of our proposed encoder is proved by the comparison with the previous works.

In Sequence to Sequence model, we consider the input document as the sequence of words $d = \{w_1, w_2, ..., w_n\}$. In our experiment, each word is mapped into the vector space by Word2Vec model [14]. The Word2Vec embedding matrix[4] we use is pre-trained on Google News dataset with about one hundred billion words. The dimension of each word vector is three hundred. With the words which have no appearance in the pre-trained matrix, they are randomly calculated by the normal distribution with $mean = 0$ and standard deviation $= \sqrt{0.25}$. In the other hands, the size of vocabulary is 20000.

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

The next part in our model is the setting for encoding and decoding phase. In our model, the encoder is the combination of ConvNet and BiLSTM. With the CNN-based encoder, we use three different channels to capture three kinds of phrases in the input sequence which includes bigram, 6-gram, and 12-gram. For each channel, there are 25 filters, and the sliding step (strides) is 1. For pooling operation, we use the max function with the size of the region is 3. After, the final state of ConvNet encoder is the concatenation of three channels which are fed into the Feed-forward Neural Network with ReLU as the activation function. The dimension of output in this network is 100. With the bidirectional LSTM encoder, the dimension of each LSTM unit is 50, hence the dimension of the final hidden state computed in Equation 2.9 is 100. Finally, the encoding hidden state is the addition of ConvNet and BiLSTM one like 3.1. In the decoding phase, we use LSTM model with the initial state as the encoding hidden one, so the dimension of LSTM unit is 100.

To adapt with CNN/Daily Mail, the limitation of the length of the document and summary is 800 and 100 tokens. If the sequence is longer than the upper bound, it is cut from the start of the sequence. In the case that the sequence is so short, we pad it by zero at the end of the sequence. With these constraints, the dimension of intra-temporal and intra-decoder attention matrix is $(800, 100)$ and $(100, 100)$. Therefore, the size of context vector which is the combination of the hidden state in the decoder and two kinds of attention in each time step t is 300. This vector is fed into the Feed-forward Neural Network to get the probability distribution of word in the vocabulary. The detail of these above settings in the form of our proposed model is presented in Table 4.2.

For the training process, each model in the experiment has the different loss function based on the goal of these evaluations. However, the following settings are used for all models. We use RMSProp with the learning rate is 0.001, and its decay is 0 for the optimization process. The number of samples in each batch is 80. After training process, the size of Beam Search[5] [13] in predicting is chosen by 3. All above information is presented in Table 4.3.

4.3 Measurements

To evaluate the Summarization system, we use the ROUGE measurement which is the most popular one in this topic. The main idea of ROUGE is based on the overlapping ratio between the generated and the target summary. In our study, we focus on ROUGE-1, ROUGE-2, and ROUGE-L as the main measurement for the system which reflects two aspects of the expected summary. Firstly, with ROUGE-N score, we want to evaluate the correctness of our system through n-gram overlap. In the other side, ROUGE-L is to evaluate the fluency of the generated summary. It comes from the assumption that if the length of common subsequence between two kinds of the summary is large, the

[5] `https://en.wikipedia.org/wiki/Beam_search`

	Size/dimension/Quantity
Input Doc	800 tokens
Summary	100 tokens
ConvNet Channel	3 (2 - 6 - 12)
ConvNet Filter	25/channel
ConvNet Hidden State	100
BiLSTM Hidden State	100
Encoder Final State	100
Decoder LSTM Unit	100
Intra-temporal Attention Matrix	(800, 100)
Intra-decoder Attention Matrix	(100, 100)
Word vector	300

Table 4.2: The detail of our proposed model's settings

generated one is more smooth and fluent. If the summary only consists the duplication of the separated n-grams without the semantic relation, ROUGE-N score is also high, but it is not useful for practice. In the experiment, we use the tool[6] developed by Chin-Yew LIN in 2005 with the same hyper-parameters[7] in the previous works.

4.4 Results

Firstly, we compare the results among three of our models which is shown in Table 4.4. In these evaluations, we can see that the quality of RConvBiLSTM is better than the others in ROUGE-1 and ROUGE-L scores while the combined system - COConvBiLSTM gets the best result on ROUGE-2 score. The reason for this phenomena is that the loss function of RConvBiLSTM is designed to focus on the ROUGE-L, which makes the result in this score so high. With the significant value of ROUGE-1, we can conclude that reinforced objective function is also effective for the generation in unigram. When we combine two loss functions in COConvBiLSTM, the result of ROUGE-1 and ROUGE-L becomes a bit bad. The reason is that the model has to optimize two targets of learning. In this sense, the model has no ability to entirely focus on the reward function

[6] `https://github.com/andersjo/pyrouge/tree/master/tools/ROUGE-1.5.5`

[7] -c 95 -2 4 -U -r 1000 -n 2 -w 1.2 -m -s

	Settings
Optimizer	RMSProp
Learning rate	0.001
Learning rate decay	0.0
Batch size	80
Beam Search	3

Table 4.3: The detail of our training and predicting process

like RConvBiLSTM. However, as we mentioned above on the strength of condition log-likelihood loss function, it is useful to increase the probability of sequence in the same way as language modeling solutions. These properties help our model with the improvement in ROUGE-2 score which is based on the bigram overlap. Therefore, the rise and the fall of those systems is the common trade-off in objective learning.

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
RConvBiLSTM	**42.45**	16.02	**39.09**
ConvBiLSTM	39.17	15.06	35.62
COConvBiLSTM	42.30	**16.25**	37.07

Table 4.4: The result of our proposed models

Next important part is to prove the robustness and effectiveness of our proposed model with the previous words in this topic. The comparison is shown in Table 4.5. The top part in this table is the Extractive Summarization model and the second one is on Abstractive task. In both of two kinds of previous works, our model is proved to be more effective in ROUGE-1 and ROUGE-L score. Especially, RConvBiLSTM model has a significant improvement in ROUGE-L. However, in ROUGE-2, our models do not outperform with Pointer-Generator Network. It comes from the effect of pointer mechanism that maybe our next inspiration.

	ROUGE-1	**ROUGE-2**	**ROUGE-L**
Lead-3[16]	39.20	15.70	35.50
SummaRuNNer[16]	39.60	16.20	35.30
SummaRuNNer-abs[16]	37.50	14.50	33.40
Pointer-Generator Net [22]	39.53	**17.28**	36.38
DReinforcedModel [18]	41.16	15.75	39.08
RConvBiLSTM	**42.45**	16.02	**39.09**
COConvBiLSTM	42.30	16.25	37.07

Table 4.5: The comparison of our models and the previous works in Summarization

4.5 Discussion

In this section, we discuss some samples in testing to reflect the effectiveness of our model. Firstly, in the case of the short summary, RConvNet and COConvNet almost produce correctly the summary. With ConvBiLSTM, in some cases, despite the high ROUGE score, the fluency is not guaranteed as Table 4.6. The short target summary is quite simple and possible to catch by the Deep Learning systems.

Input Document (ID: 10316 - Test set)
johnny depp has one of the most distinctive faces in hollywood , but he 's almost unrecognizable in the first trailer for **his latest film black mass** . in the long-awaited trailer for the film **to be released this september** , depp plays james white ' bulger - the notorious boston gangster who topped the fbi 's most wanted list for 16 years before he was finally arrested in 2011 ...
Ground Truth
black mass is set to be released in september 2015 .
RConvBiLSTM (ROUGE-1: 90.9)
black mass is set to be released this september
COConvBiLSTM (ROUGE-1: 90.9)
black mass is set to be released this september
ConvBiLSTM (ROUGE-1: 66.67) latest film black mass to be released this september

Table 4.6: The example for the short target summary

However, in the case of long target summary, the automatic summarization is too difficult. The reason is that the semantic meaning of the ground truth sequence is not concerned. This sequence is just divided into the word's elements and use for the training process as the label. However, in this case, the quality of our generated summary is also acceptable like the example in Table 4.7. The common error we observe is the duplication of information in the summary in unigram and bigram which leads to the lack of space for prediction because of our upper bound of the summary is 100 tokens.

Input Document (ID: 869 - Test set)
sao paulo , brazil throngs of protesters packed the streets of major brazilian cities on sunday , **pushing for the impeachment of president dilma rousseff** . fueled by mounting anger over **a corruption scandal** that has implicated politicians **in rousseff 's party** , demonstrators chanted out with dilma and time for change . " **police estimated that 275,000 demonstrators marched in sao paulo** . a sea of protesters dressed in the green and yellow of the brazilian flag used decades-old rallying cries to fire up their ranks , singing rock songs that date back to protests of the country 's one-time military dictatorship . some protesters said they 'd rather see **rousseff step down than push for impeachment** , which could be difficult to push through without evidence tying the president directly to the corruption scandal . but janaina said impeachment remained a realistic option . ' yes , it has to be ' she said . ' it 's our last hope . '
Ground Truth
police say 275,000 demonstrators marched in sao paulo . many want president dilma rousseff to be impeached . a corruption scandal has implicated politicians in her party .
RConvBiLSTM (ROUGE-1: 64.7)
a corruption scandal in rousself's party pushing for the impeachment of president . police estimated that NUMBER demonstrators marched in sao paulo . rousseff step down than push for impeachment
COConvBiLSTM (ROUGE-1: 57.89)
police estimated NUMBER demonstrators marched in sao paulo . some protesters said rather see rousseff step down than push for impeachment pushing for the impeachment of president dilma rousseff fueled by mounting anger over a corruption scandal
ConvBiLSTM (ROUGE-1: 45.16) president dilma roussef to be impeached roussef step down than push for impeachment impeachment police estimated that NUMBER demonstrators in sao paulo

Table 4.7: The example for the long target summary

Chapter 5

Conclusion and Future Works

In the inspiration of Reinforced Mechanism in Abstractive Summarization, we propose a lot of combination of techniques to solve some central problems in this task. From many experiments we have done, we consider that RConvBiLSTM obtains the best results in ROUGE-1 and ROUGE-L measurement. In the other hands, with the complexity of our experimental proposed model, these models are possible to apply into practice, which is proved in Chapter 4 with more than two hundred thousands of samples for training. Experimental results demonstrate that the combination of ConvNet and BiLSTM is more meaningful and useful than the previous encoder model for Abstractive Summarization. Generally, our model outperforms the current state-of-the-art model by 1.3% for ROUGE-1 and 0.4% in ROUGE-2. It comes from our contribution on designing the novel architecture based on the smooth combination of the different networks and mechanism. Simultaneously, these integrations are also inspired by the latest technology in text understanding problem.

To prove the effectiveness of our model in Abstractive Summarization, we will take more experiments in the other datasets. Those results and observation is the good instruction in our future work to improve the internal problem in our system. For example, through the discussion part in Chapter 4, we observe that the role of the target summary in both our models and the previous works is still quite limited in the treatment of labels for prediction. Therefore, in the next step, we will take into account to modify its role in the learning process. It should be considered as the semantic constraints for learning.

Bibliography

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

[2] Regina Barzilay and Kathleen R. McKeown. Sentence fusion for multidocument news summarization. *Comput. Linguist.*, 31(3):297–328, September 2005.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994.

[4] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Sanda Harabagiu Finley and Sanda M. Harabagiu. Generating single and multi-document summaries with gistexter. In *In U. Hahn and D. Harman (Eds.), Proceedings of the workshop on automatic summarization*, pages 30–38, 2002.

[6] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348. Association for Computational Linguistics, 2010.

[7] Pierre-Etienne Genest and Guy Lapalme. Fully abstractive approach to guided summarization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 354–358, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[8] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA, 2015. MIT Press.

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

[10] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014.

[11] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

[12] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics, 2015.

[13] M.F. Medress, F.S. Cooper, J.W. Forgie, C.C. Green, D.H. Klatt, M.H. O'Malley, E.P. Neuburg, A. Newell, D.R. Reddy, B. Ritea, J.E. Shoup-Hummel, D.E. Walker, and W.A. Woods. Speech understanding systems: Report of a steering committee. *Artificial Intelligence*, 9(3):307 – 316, 1977.

[14] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[15] N. Moratanch and S. Chitrakala. A survey on abstractive text summarizations. In *Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–7, 2016.

[16] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *Proceedings of the 31st AAAI Conference*, page 7, 2017.

[17] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics, 2016.

[18] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *Sixth International Conference on Learning Representations (ICLR)*, 2018.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[20] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1179–1195, 2017.

[21] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.

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

[23] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

[24] Dingding Wang and Tao Li. Weighted consensus multi-document summarization. *Inf. Process. Manage.*, 48(3):513–523, May 2012.