QUERY FOCUSED ABSTRACTIVE SUMMARIZATION USING BERTSUM MODEL

DEEN MOHAMMAD ABDULLAH
Master of Science, Bangladesh University of Engineering and Technology, 2014
Bachelor of Science, Military Institute of Science and Technology, 2010

A thesis submitted
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

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

QUERY FOCUSED ABSTRACTIVE SUMMARIZATION USING BERTSUM MODEL

DEEN MOHAMMAD ABDULLAH

Date of Defence: August 14, 2020

Dr. Yllias Chali Thesis Supervisor	Professor	Ph.D.
Dr. Wendy Osborn Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. John Zhang Thesis Examination Committee Member	Associate Professor	Ph.D.
Dr. John Sheriff Chair, Thesis Examination Committee	Assistant Professor	Ph.D.

Dedication

Dedicated to Prophet Hazrat Muhammad (SM.)

Abstract

In Natural Language Processing, researchers find many challenges on Query Focused Abstractive Summarization (*QFAS*), where Bidirectional Encoder Representations from Transformers for Summarization (*BERTSUM*) can be used for both extractive and abstractive summarization. As there is few available datasets for *QFAS*, we have generated queries for two publicly available datasets, CNN/Daily Mail and Newsroom, according to the context of the documents and summaries. To generate abstractive summaries, we have applied two different approaches, which are Query focused Abstractive and Query focused Extractive then Abstractive summarizations. In the first approach, we have sorted the sentences of the documents from the most query-related sentences to the less query-related sentences, and in the second approach, we have extracted only the query related sentences to fine-tune the *BERTSUM* model. Our experimental results show that both of our approaches show good results on ROUGE metric for CNN/Daily Mail and Newsroom datasets.

Acknowledgments

Bismillahir Rahmanir Rahim (In the name of Allah, the Most Gracious, the Most Merciful)

First of all, I would like to thank the almighty Allah for granting me to complete this work for my M.Sc. degree from the University of Lethbridge and living abroad with his mercy.

I am very grateful to Professor Dr. Yllias Chali for his endless support to achieve my goal. He was not only my mentor but also made me feel that he is my guardian in abroad.

I want to thank my M.Sc. supervisory committee members Dr. Wendy Osborn, and Dr. John Zhang for their time, effort, and valuable suggestions to my research.

I am grateful to all my mentors from whom I took courses and gave me a chance to share a part of their immense knowledge.

I also have to thank the University of Lethbridge for the financial support.

I am grateful to my parents (Dr. Mohammad Mohiuddin Abdullah and Afsir Begum) for always encouraging me about my higher study. Their endless supports and sacrifices help me to move forward in my life. I am also thankful to my siblings, Mahmuda and Wali.

Last but not the least, I would like to thank my wife (Sara Binte Zinnat) and daughter (Shifa Abdia Deen) for being with me in my hardship abroad. Their love and support encourage me to continue my study with the Ph.D. program.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Natural Language is one of the most sophisticated features of human knowledge (Christiansen and Chater, 2008). As a human, we have different natural languages to communicate with people from different cultures, customs, and countries. Natural languages have been processed by computational methods to analyze the data, most commonly the textual data such as documents in the field of Natural Language Processing (*NLP*) (Doszkocs, 1986). In the modern era, information can be accessed using the internet. Still, it is time-consuming to go through single or multiple documents to get proper information. Researchers are working towards providing summarized information instead of large texts to avoid missing the vital information of that document within a short time. Hence, in *NLP*, text summarization has become an interesting area for the researchers.

There are two types of text summarization: extractive summarization and abstractive summarization. In the extractive summarization, salient sentences are selected to generate extractive summaries. On the other hand, in the abstractive summarization, sentences are paraphrased to generate abstractive summaries. Extractive summaries may lose the main context of the documents whereas in abstractive summaries, one can get the actual context of the document. But the main challenge to generate abstractive summaries is to create grammatically correct sentences. Sometimes, instead of providing the whole context of the document, people may need a partial contextual summary from the document according to their given query, which is known as query focused summarization. For example, a

user may need to know only the weather forecast for the City of Lethbridge instead of the weather summaries of the whole of Canada.

Recently, only a few researchers have shown their interest in generating abstractive summaries based on relevant queries (Nema et al., 2017; Hasselqvist et al., 2017; Baumel et al., 2018; Lierde and Chow, 2019; Egonmwan et al., 2019). Abstractive summarization has a research challenge to generate grammatically correct sentences. In Query Focused Abstractive Summarization (*QFAS*), the shortage of appropriate datasets has created the research challenges for the researchers. In this research, we are motivated to take this research challenge and accomplish our research on *QFAS*. Pretrained language model is a black box that understands the language and can then be asked to do any specific task. We can fine-tune a pretrained language model with our selected datasets instead of building a model from scratch. Using a pretrained language model, we can get better abstractive summaries. Liu and Lapata (2019) have developed a pretrained language model, namely *BERTSUM* for extractive and abstractive summarizations. Therefore, we hypothesize that incorporating the query into the *BERTSUM* model can provide improved performance over current state-of-the-art results.

1.2 Contribution

- We have generated queries for two publicly available datasets, namely CNN/Daily Mail (Hermann et al., 2015), and Newsroom (Grusky et al., 2018) considering the context of the documents and summaries. The generation and incorporation of queries for two public datasets are one of our contributions that have helped us to implement our proposed approaches for generating query focused summaries.

- We have implemented the existing *BERTSUM* model (Liu and Lapata, 2019) to generate abstractive summaries. In this approach, the query was not considered, which motivated us to propose two approaches for incorporating queries in the *BERTSUM* model.

- We have introduced a novel approach, which is *QAbs* (query focused abstractive) summarization, where we have sorted all the sentences of the document according to the query and then used them to fine-tune the *BERTSUM* model to generate abstractive summaries.

- We have also proposed a unique approach, which is *QExAbs* (query focused extractive then abstractive) summarization, where we have selected the query-related sentences and then used them to fine-tune the *BERTSUM* model to generate abstractive summaries.

- We further implemented and evaluated these two approaches, *QAbs* and *QExAbs*, with other existing abstractive summarization models.

1.3 Overview of the Thesis Organization

We organize the rest of this thesis as follows. In Chapter 2, we have presented the literature review and background of this thesis. Our query generation approach has been discussed in Chapter 3. Chapter 4 describes our use of a pretrained language model, namely *BERTSUM* to generate abstractive summaries without considering the query. Chapter 5 illustrates our first unique approach: query focused abstractive summarization (*QAbs*), and Chapter 6 explains our another novel approach: query focused extractive then abstractive summarization (*QExAbs*). Finally, Chapter 7 concludes the thesis and proposes directions for future research.

Chapter 2

Background

Automatic text summarization has become an area of interest for researchers for the last sixty years (Gambhir and Gupta, 2017). In this chapter, we have described some preliminaries and background of our research.

2.1 Artificial Neural Network

An Artificial Neural Network (*ANN*) is a biologically inspired computational network. *ANN* has become the primary tool of Machine Learning and capable of performing extensively parallel computations for data processing and knowledge representation (Schalkoff, 1997). The elementary unit of an *ANN* is an artificial neuron or perceptron. Several perceptrons used together work like a human brain.

2.1.1 Perceptron

The fundamental processing unit of a neural network is a neuron or perceptron. It works like a biological neuron that can receive input from other sources and combines them by performing a nonlinear operation to generate and output the result. The relationship between the essential parts of a biological neuron is shown in Figure 2.1.

The algorithm of the artificial neuron (perceptron) has been developed following the four necessary parts of the biological neuron. The basic structure of a perceptron is shown in Figure 2.2. This example describes an $(n+1)$ input perceptron where each input has some weight w_i. After multiplying each input x_i with their weight w_i, all the results are

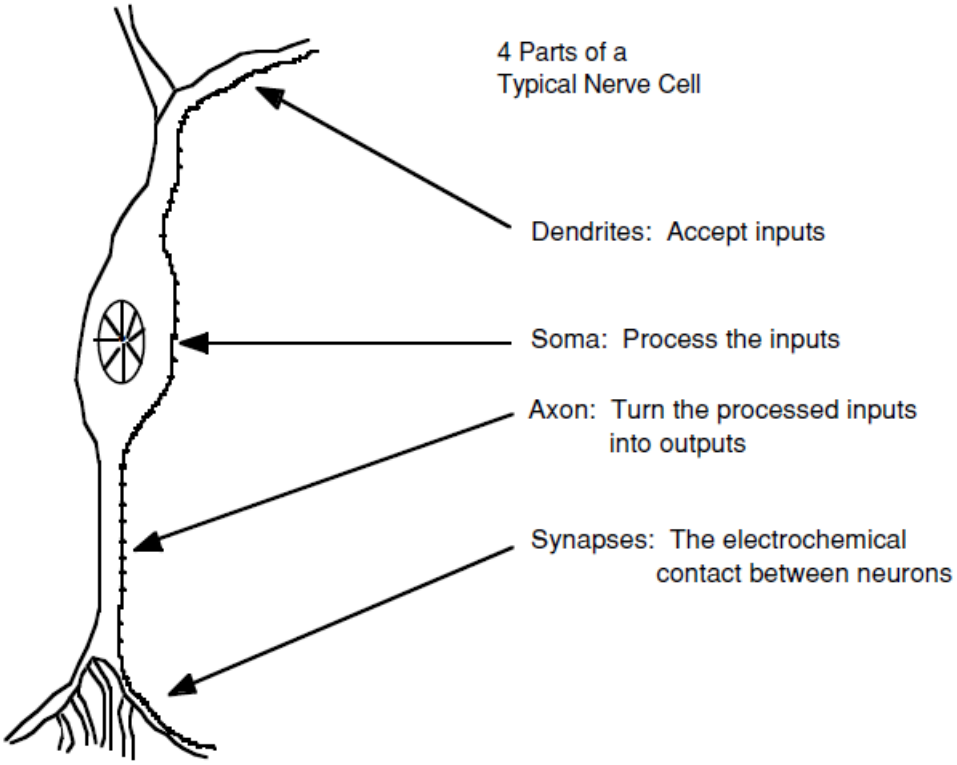


Figure 2.1: A simple biological neuron (Anderson and McNeill, 1992)

summed. In the final step, a transfer or activation function decides whether the perceptron should be activated or not.

In practice, a bias value is added with the multiplied results of all inputs x_i and their weights w_i of a perceptron shown in Equation 2.1.

$$z = (\sum_{i=0}^{n} w_i x_i) + b \tag{2.1}$$

For the activation or transfer function, the sigmoid function is commonly used, as shown in Equation 2.2.

$$y = \sigma(z) = \frac{1}{1+e^{-z}} \tag{2.2}$$

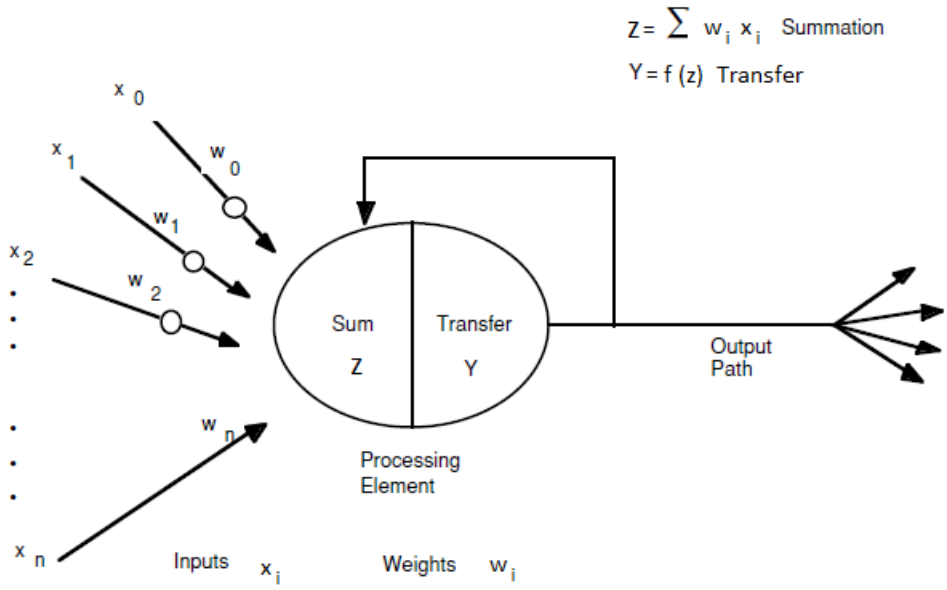


Figure 2.2: Perceptron: An artificial neuron (Anderson and McNeill, 1992)

2.1.2 Building the network

Several perceptrons are connected to build the *ANN*, where these perceptrons are arranged in a series of layers. In *ANN*, there are three types of layers: the input layer, hidden layers, and the output layer. Here, the number of input and output layer should be one, but the number of hidden layers can be zero or more. A basic structure of *ANN* is shown in Figure 2.3.

2.1.3 Recurrent Neural Network (*RNN*)

A recurrent neural network (*RNN*) is different from traditional *ANN*, which can use previous states of the network to generate better output. The structure of the *RNN* is shown in Figure 2.4, where x_1 and x_2 are the inputs, and y is the output. Long Short-Term Memory (*LSTM*) and Gated Recurrent Units (*GRUs*) are the variants of *RNN*.

***LSTM*:** Hochreiter and Schmidhuber (1997) introduced *LSTM* which is an improved version of *RNN*. A common *LSTM* unit is composed of a cell, an input gate, an output gate and a forget gate.

***GRUs*:** Cho et al. (2014) introduced *GRUs* by incorporating gating mechanism in *RNN*.

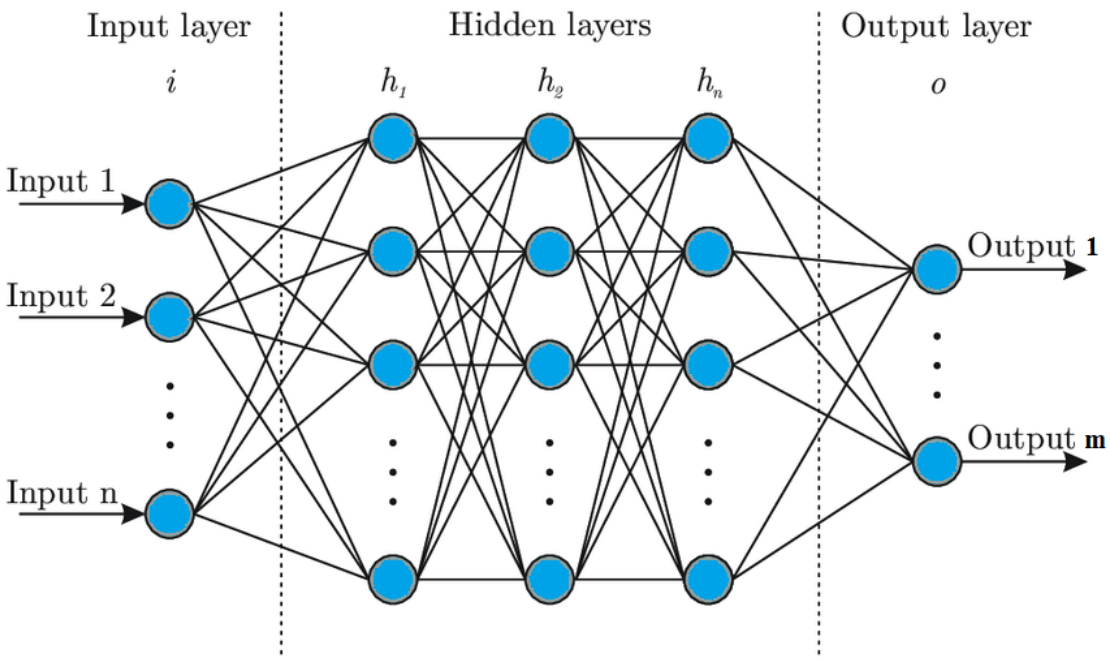


Figure 2.3: Artificial Neural Network (*ANN*) with hidden layers (Bre et al., 2018)

The *GRU* is pretty similar to an *LSTM*. *GRUs* got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate.

2.1.4 Feed-Forward neural network

A particular type of *ANN* is the Feed-forward neural network (*FFN*), where the decision flow is unidirectional. The connection from the input layer to the output layer has several hidden layers with no cycles or loops (Schmidhuber, 2015).

2.2 Sequence-to-Sequence Model

A Sequence-to-Sequence (*seq2seq*) model can be used for different applications such as speech recognition, video to text conversion, classification, question answering, and text summarization (Dong et al., 2018; Venugopalan et al., 2015; Tang et al., 2016; He et al., 2017; Liu et al., 2018).

Generally, the *seq2seq* model converts one sequence into another sequence. In text summarization, a *seq2seq* model takes a sequence of words or sentences as input and predicts

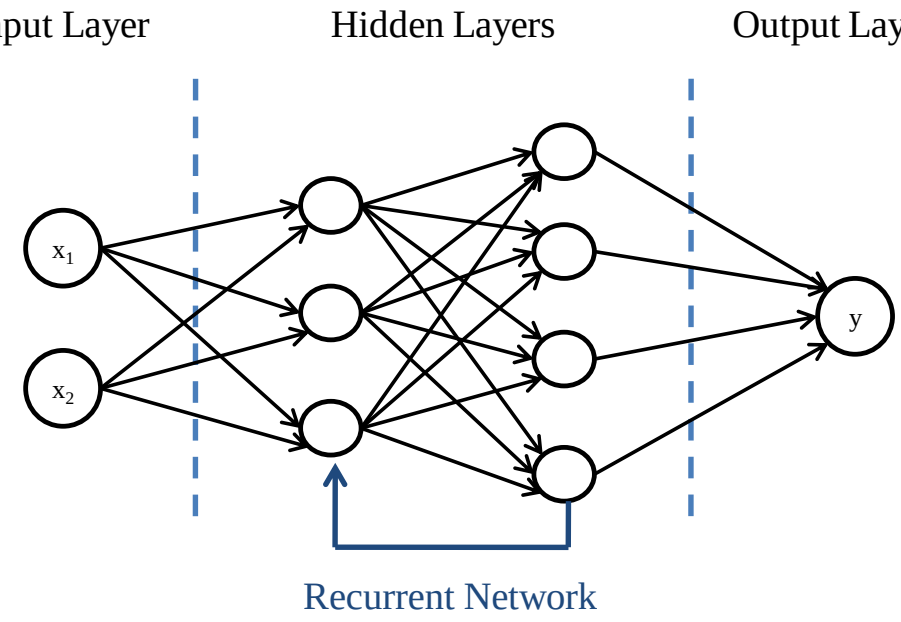


Figure 2.4: Recurrent Neural Network (*RNN*)

a sequence of words or sentences as output. *Seq2Seq* model uses a *RNN*. Instead of using the basic version of the *RNN*, the *seq2seq* model uses more advanced versions: *LSTM* and *GRUs* to solve the problem of the vanishing gradient where the vanishing gradient problem happens for training the model with gradient-based learning methods and backpropagation. The *seq2seq* model has two parts: Encoder and Decoder where both of them are a collection of several recurrent units (*LSTM* or *GRUs*).

Encoder: The encoder creates a smaller dimensional representation of the input to understand the input sequence and then forwards that representation to the decoder.

Decoder: The decoder generates a sequence of its own that represents the output.

The basic architecture of a *seq2seq* model is shown in Figure 2.5.

2.3 Transformer Model

Vaswani et al. (2017) introduced the transformer model based on attention mechanisms. Since the introduction of the attention mechanism based transformer model, the model has

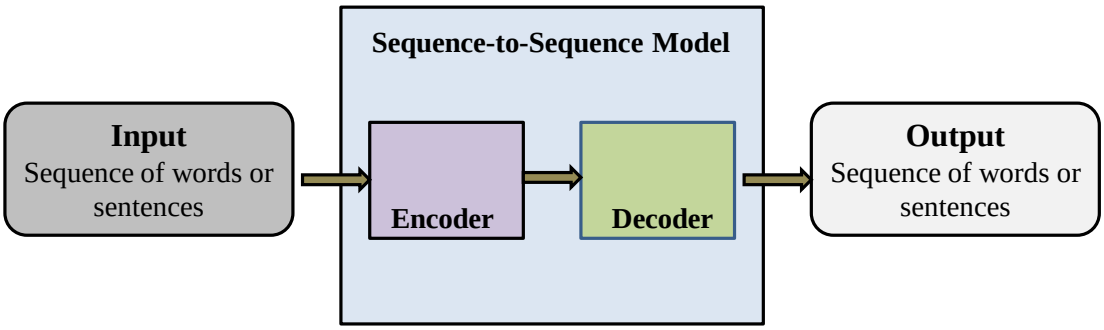


Figure 2.5: Seq2Seq: Encoder-Decoder Model

become an essential building block of many state-of-the-art models in Natural Language Processing (*NLP*). Recently, Egonmwan and Chali (2019b) have combined both the transformer and sequence-to-sequence models for summarization. In other work, Egonmwan and Chali (2019a) have used the transformer model to capture long-term dependencies. The transformer model has the following essential parts: Encoder and Decoder Stacks, Attention, Position-wise Feed-Forward Networks, Embeddings followed by Softmax, and Positional Encoding. A basic architecture of transformer model is shown in Figure 2.6.

2.3.1 Encoder and Decoder Stacks

In the transformer model, the encoder has a stack of six identical layers where each layer contains two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Vaswani et al. (2017) applied a residual connection (He et al., 2016) around each of the two sub-layers, followed by layer normalization (Lei Ba et al., 2016). From each sub-layer, the generated output is shown in Equation 2.3. Here, $Sublayer(x)$ is the function implemented by the sub-layer itself. All the sub-layers, including the embedding layers, produce outputs of dimension $d_{model} = 512$.

$$Output_{SubLayer} = LayerNorm(x + Sublayer(x)) \tag{2.3}$$

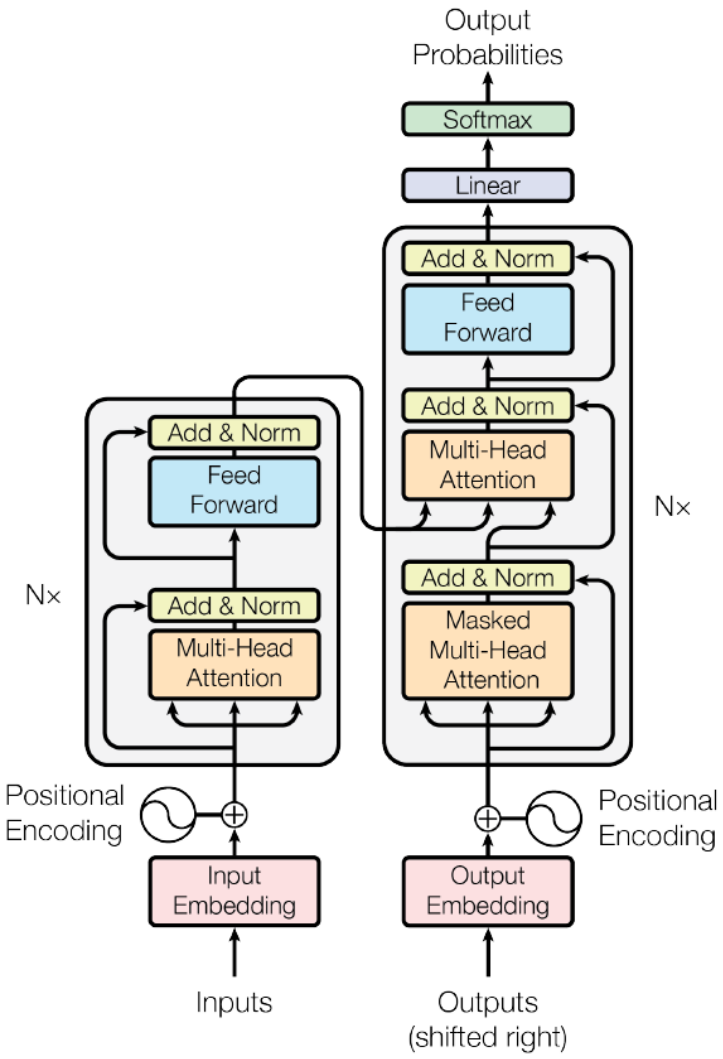


Figure 2.6: Architecture of Transformer Model (Vaswani et al., 2017)

Like the encoder, the decoder also has a stack of six identical layers where each layer contains three sub-layers. Among three sub-layers, two layers are similar to the encoder, where one additional layer has been incorporated to perform multi-head attention over the output of the encoder stack. Residual connection around three sub-layers, followed by the layer normalization, has been implemented the same as the encoder. The self-attention sub-layer has been modified in the decoder by masking positions to prevent them from attending the following positions, which helps to predict position i depending on the outputs at positions less than i.

2.3.2 Attention

Mapping the vectors of a query (Q) and a set of key-value pairs (K, V) to output is done by an attention function. The generated output is the addition of the weighted values assigned to the queries and the corresponding key-values pairs.

Scaled Dot-Product Attention shown in Figure 2.7(a) is used to design the Multi-Head Attention shown in Figure 2.7(b) in the transformer model. In Scaled Dot-Product Attention, the input consists of queries and keys of dimension d_k and values of dimension d_v. To calculate the weights of the values, the dot product of the queries and the keys are divided by $\sqrt{d_k}$ and then passed to a softmax function. Finally, the output of the attention is generated, as shown in Equation 2.4, where queries, keys, and values are represented as matrices Q, K, and V, respectively.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4}$$

In Multi-Head Attention, queries, keys, and values are linearly projected in h parallel attention layers, as shown in Equation 2.5. Here, $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^0 \in \mathbb{R}^{hd_v \times d_{model}}$ (Vaswani et al., 2017). In practice, $h = 8$ and hence, $d_k = d_v = d_{model}/h = 64$.

$$MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O \tag{2.5}$$

In the transformer, the multi-head attention has been used in three different ways. In one way, queries come from the previous decoder layer, whereas keys and values come from the encoder's output. All of the keys, values, and queries come from the previous layer's output in the encoder as a second approach. Otherwise, each position in the decoder is allowed to attend to all positions.

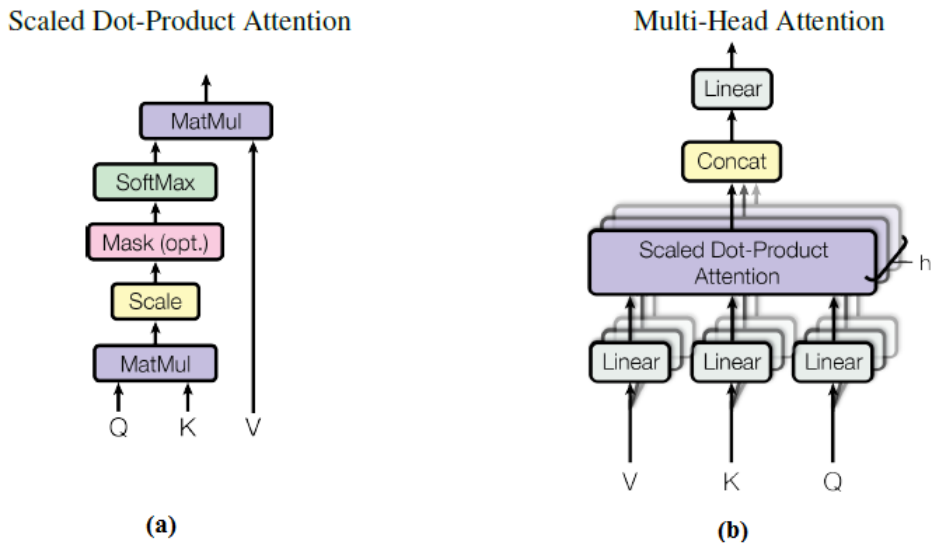


Figure 2.7: (a) Scaled Dot-Product Attention. (b) Multi-Head Attention consists of several attention layers running in parallel. (Vaswani et al., 2017)

2.3.3 Position-wise Feed-Forward Networks

The transformer model has a fully connected Feed-Forward Network (*FFN*) applied to each position independently and identically.

2.3.4 Embeddings and Softmax

Vaswani et al. (2017) used embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. Both linear transformation and the softmax function (Press and Wolf, 2017) have been used to convert the decoder output to predict next-token probabilities.

2.3.5 Positional Encoding

On the bottom of both the encoder and decoder stacks, position embedding (Gehring et al., 2017) is added to the transformer model. The author used the sine and cosine functions of different frequencies for position encoding in the transformer, as shown in Equation

2.6 and Equation 2.7. Here, *pos* is the position, *i* is the dimension ($i \in [0, 255]$).

$$PE(pos, 2i) = sin(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.6}$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.7}$$

For example, if $d_{model} = 4$, then we will get a 4-dimensional vector as follows which can be added with a 4-dimensional input to generate position embedding of that input:

$$[sin(\frac{pos}{10000^0}), cos(\frac{pos}{10000^0}), sin(\frac{pos}{10000^{(2/4)}}), cos(\frac{pos}{10000^{(2/4)}})]$$
$$= [sin(pos), cos(pos), sin(\frac{pos}{100}), cos(\frac{pos}{100})]$$

2.4 Pretrained Language Models

We can use a pretrained model with fine-tuning the model with our dataset instead of building a model from scratch to solve a similar *NLP* problem. By using a pretrained model, we can reduce the computation time and production costs. There are several pretrained models available for *NLP*, but we have to select an appropriate pretrained model by which we can fine-tune on our dataset.

2.4.1 BERT

Bidirectional Encoder Representations from Transformers (*BERT*) (Devlin et al., 2019) is developed by Google. This model is designed by jointly conditioning on both left and right context in all layers to pre-train deep bidirectional representations from the unlabeled text. *BERT* will be described in detail in Section 2.5.

2.4.2 GPT

Generative Pre-Training (*GPT*) is the pretrained language model on a diverse corpus of unlabeled text developed by OpenAI (Radford et al., 2018). Task-aware input transfor-

mations are used to achieve effective transfer during fine-tuning, which requires minimal changes to the model architecture.

2.4.3 GPT-2

Radford et al. (2019) claimed that their *GPT-2* model can perform in a zero-shot setting. The zero-shot setting is a problem setup in machine learning, where the model tests the data without observing the predicted class during the training. As the model is trained to maximize the likelihood of a sufficiently varied text corpus, it learns how to perform a remarkable amount of tasks without the need for explicit supervision.

2.4.4 Transformer-XL

Transformer-XL has been developed by Google/Carnegie Mellon University(CMU)[1], which enables learning dependency beyond a fixed-length without disrupting temporal coherence (Dai et al., 2019), where learning visual invariance is called the temporal coherence. *Transformer-XL* model has been built using a segment-level recurrence mechanism with a novel positional encoding scheme that solves the context fragmentation problem. Context fragmentation refers to when the model lacks the necessary contextual information to predict the first few symbols due to the way the context was selected.

2.4.5 XLNet

XLNet has also been developed by Google/Carnegie Mellon University(CMU)[2], which is a generalized autoregressive pretraining method (Yang et al., 2019). This model can enable learning bidirectional contexts by maximizing the expected likelihood and overcoming the limitations of *BERT*. The *BERT* predicts the masked tokens independently, so it does not learn how they influence one another.

[1] https://github.com/kimiyoung/transformer-xl
[2] https://github.com/zihangdai/xlnet/

2.4.6 XLM

The cross-lingual language model (*XLM*) has been released by Facebook, where the authors proposed two methods for the learning phase (Lample and Conneau, 2019). In one way, they used an unsupervised approach that relies on monolingual data. In the other method, the authors used a supervised method that leverages parallel data with a new cross-lingual language model objective.

2.4.7 RoBERTa

Liu et al. (2019) have modified the pretraining procedure of *BERT* to improve end-task performance. The authors have trained their model longer, with bigger batches over more data, and removed the next sentence prediction objective. In their approach, training has been done on longer sequences.

2.4.8 DistilBERT

Sanh et al. (2019) proposed a general-purpose pre-trained version of BERT where the authors compressed the *BERT* model into a small model. The authors claimed that their model is smaller, faster, cheaper, and lighter.

2.4.9 CTRL

Keskar et al. (2019) have released a 1.63 billion-parameter conditional transformer language model, *CTRL*. It provides a potential method for analyzing large amounts of generated text by identifying the most influential source of training data in the model. They claimed that human users could more easily control text generation using their model.

2.4.10 CamemBERT

Based on the *RoBERTa* model, Martin et al. (2019) have trained their model using the French language. They evaluated four downstream tasks: part-of-speech tagging, dependency parsing, named entity recognition, and natural language inference. Downstream tasks

are those supervised-learning tasks that utilize a pre-trained model or component.

2.4.11 ALBERT

From Google Research, two optimization techniques have been proposed to lower memory consumption and increase the training speed of *BERT* (Lan et al., 2019). The two optimization techniques are:

- a factorization of the embedding layer.
- parameter-sharing across the hidden layers of the network.

2.4.12 XLM-RoBERTa

Conneau et al. (2019) have introduced a multilingual masked language model from Facebook AI. This model has been trained on 2.5 TB of newly created clean CommonCrawl (Wenzek et al., 2019) data in 100 languages. The model has shown state-of-the-art results on classification, sequence labeling, and question answering.

2.4.13 FlauBERT

FlauBERT model has learned on an extensive and heterogeneous French corpus to diverse NLP tasks (Le et al., 2019). The authors have released preprocessing and training scripts to make the pipeline reproducible.

2.5 BERT Model

Devlin et al. (2019) have developed the *BERT* model, where their framework has two steps: pre-training and fine-tuning. Model is first trained on unlabeled data over different pre-training tasks during the first phase. All the parameters are initialized and fine-tuned in the second phase. Using *BERT*, we can perform many *NLP* tasks. In Figure 2.8, a question-answering example has been shown. Here, the same architectures are used in both pre-training and fine-tuning except the output layers. For different downstream tasks, the

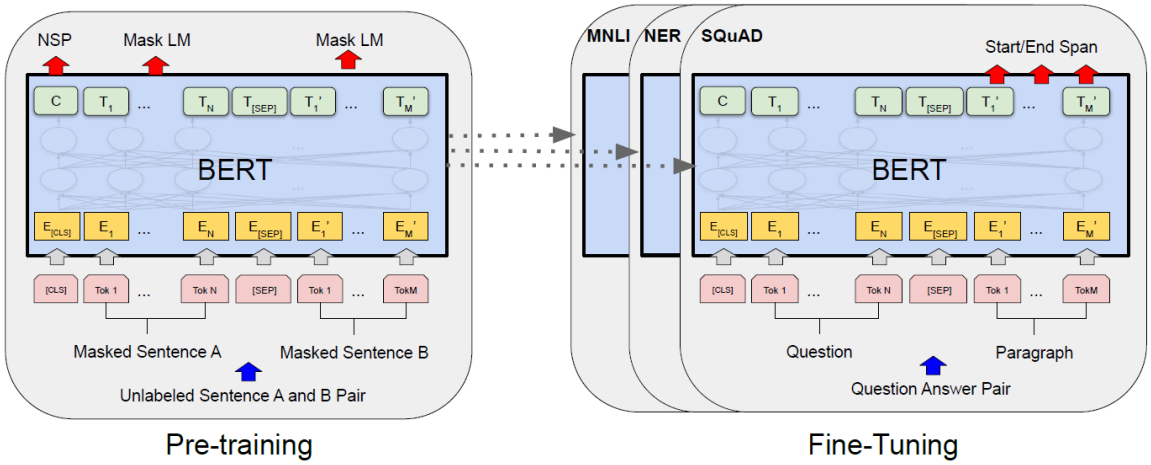


Figure 2.8: Overall pre-training and fine-tuning procedures for *BERT*. (Devlin et al., 2019)

authors used the same parameters used in pre-training to initialize the model. The authors used two special tokens $[CLS]$ and $[SEP]$ to identify the input sequences and sentences individually. The $[CLS]$ token is used to separate the input documents by appending the token at the beginning of each document. The $[SEP]$ token is used to separately identify each sentence of the document by placing the token at the end of each sentence.

2.5.1 Model Architecture and Data Representation

Based on the basic implementation of Transformer (Vaswani et al., 2017), Devlin et al. (2019) have designed a multi-layer bidirectional Transformer encoder in the *BERT* model. The number of layers (L), the hidden size (H), and the number of self-attention heads (A) are customizable in the *BERT* model.

In one token sequence, *BERT* can represent both a single sentence and a pair of sentences as input representation. The authors have used $30,000$ token vocabularies from WordPiece (Wu et al., 2016). As a sentence pair packed together into a single sequence, a token $[CLS]$ is used at the beginning of each sequence to be separated from other sequences. In Figure 2.8, the authors have used the $[SEP]$ token to identify the starting and end of each sentence along with a learned embedding for every token indicating whether it belongs to sentence A or sentence B. The input embedding has been denoted as E, the final hidden

vector of $[CLS]$ token as $C \in \mathbb{R}^H$, and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$. The input representation of the *BERT* model is shown in Figure 2.9. Here, the input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings.

2.5.2 Pre-training BERT

To pre-train the *BERT* model, Devlin et al. (2019) have used two unsupervised tasks: Masked Language Model and Next Sentence Prediction. In Masked Language Model, the authors have masked some percentage of the input tokens randomly, then predicted those masked tokens. This process can be called a *Cloze* task (Taylor, 1953). Following the standard Language Model, the corresponding masked tokens of the final hidden vectors are fed into a softmax function. The authors have predicted only the masked words instead of reconstructing the entire input for denoising auto-encoders (Vincent et al., 2008). The Next Sentence Prediction task is done by understanding the relationship between sentences. For example, choosing the sentences A and B, 50% of the time B is the actual next sentence that follows A (labeled as *isNext*), and 50% of the time, it is a random sentence from the corpus (labeled as *notNext*). In Figure 2.8, C is used to represent the Next Sentence Prediction task.

All the parameters are transferred to initialize the end-task model parameter in the *BERT* model instead of passing only the sentence embeddings (Jernite et al., 2017; Logeswaran and Lee, 2018). The authors have used 800M words from the BooksCorpus (Zhu et al., 2015) and 2,500M words from the English Wikipedia for the pre-training task.

2.5.3 Fine-tuning BERT

To Fine-tune the *BERT*, Devlin et al. (2019) have used the Transformer self-attention mechanism, which allows *BERT* to model many downstream tasks. *BERT* unifies the encoding with self-attention effectively by including bidirectional cross attention between two sentences. The authors have plugged in the task-specific inputs and outputs into *BERT* and

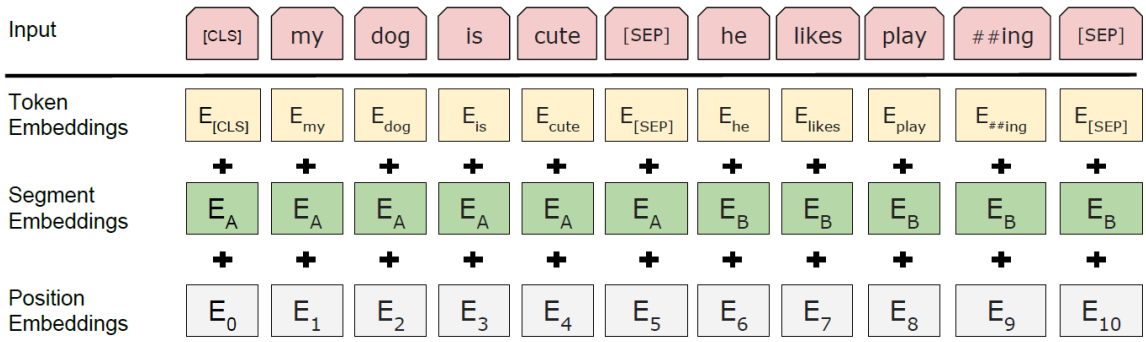


Figure 2.9: *BERT* input representation. (Devlin et al., 2019)

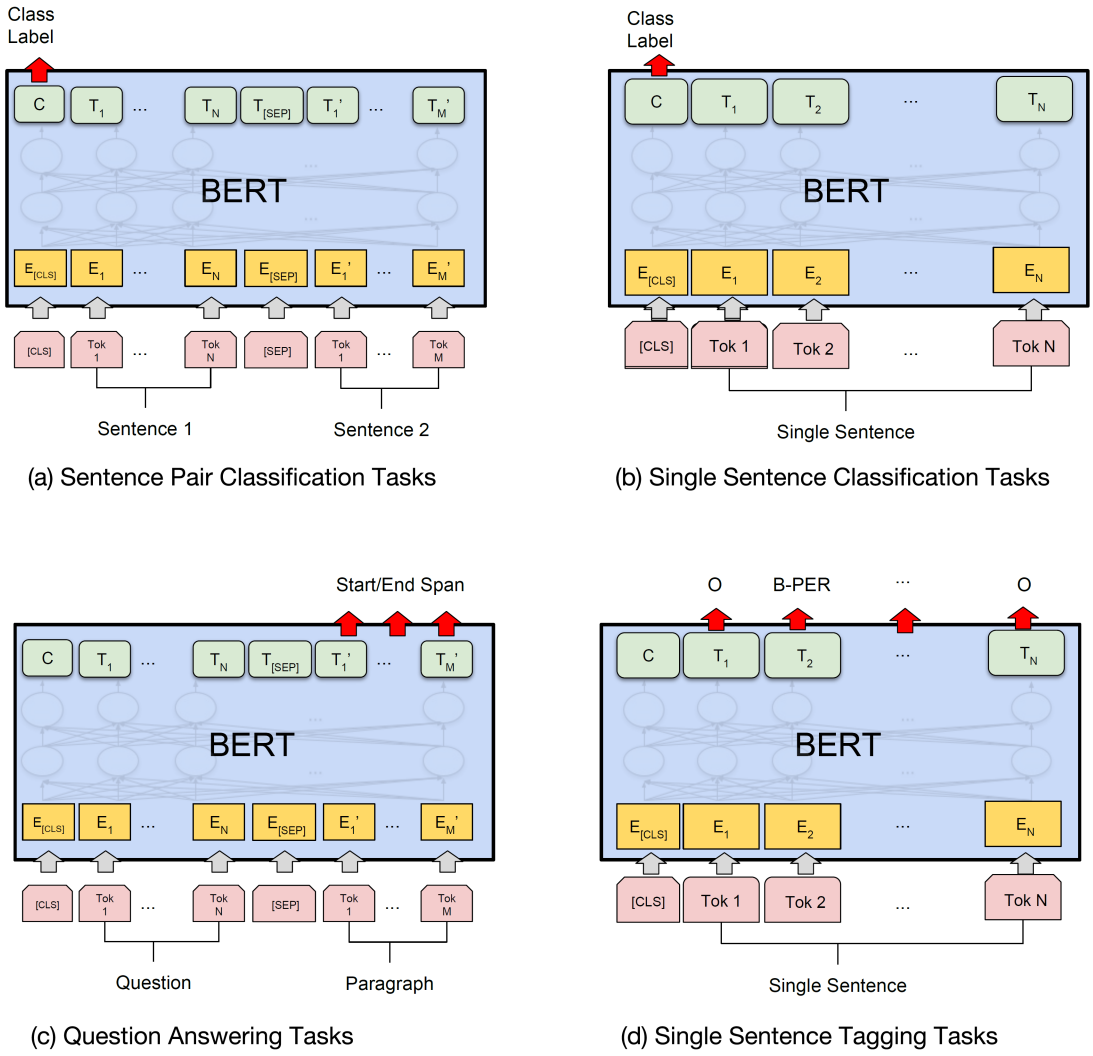
fine-tuned all the parameters end-to-end. Sentence *A* and sentence *B* from pre-training are analogous at the input, and at the output, the token representations are fed into an output layer for token level tasks. Fine-tuning for different tasks is shown in Figure 2.10. The task-specific models are formed by incorporating *BERT* with one additional output layer. Here, a minimal number of parameters need to be learned from scratch. In Figure 2.10, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks.

2.6 BERT Word Embedding

To generate high-quality feature input from the text, we can use *BERT* word embedding. Recently, neural word embeddings such as Word2Vec and Fasttext are used where vocabulary words are matched against the fixed-length feature embeddings. In *BERT*, word representations are dynamically informed by the words around them. For example, given two sentences:

"I want to deposit cash in the bank." "He loves to walk by the bank of the river."

In both sentences, the word 'bank' would have the same word embedding in Word2Vec, while under *BERT*, the word embedding for 'bank' would be different for each sentence. To obtain better model performance, generating better feature representations is essential, and hence we can use *BERT* for word embedding.



(a) Sentence Pair Classification Tasks

(b) Single Sentence Classification Tasks

(c) Question Answering Tasks

(d) Single Sentence Tagging Tasks

Figure 2.10: Illustrations of Fine-tuning *BERT* on Different Tasks. (Devlin et al., 2019)

2.6.1 Formatting the text input

BERT implicitly formats the text input into the model's expected format by incorporating some particular tokens. After tokenizing the sentences, the *BERT* model converts each token into token IDs using the *BERT* tokenizer. Mask IDs are used to distinguish between tokens and padding elements in the sequence. Segment IDs separate sentences, and position embeddings help to show the token position within the sequence. The $[CLS]$ token is used at the beginning of each sequence to be separated from other sequences. To identify the starting and end of each sentence, the $[SEP]$ token is used. For example, a given sequence:

'Bangladesh is a beautiful country. I love my motherland.'

After incorporating two special tokens, we will get following sequence:

'[*CLS*] Bangladesh is a beautiful country. [*SEP*] I love my motherland. [*SEP*]'

BERT has its own tokenizer to tokenize the sentences. Some words can be splited into subwords. For example, a given sentence:

"This is a test embedding."

This sentence can be tokenzed as:

['[*CLS*]', 'this', 'is', 'a', 'test', 'em', '##bed', '##ding', '.', '[*SEP*]']

Here, we can observe that 'embedding' has been split into subwords: ['em', '##bed', '##ding']. As the *BERT* tokenizer model contains all English characters with $30,000$ most common words found in English, the model first checks if the whole word is in the vocabulary. If the whole word is not found in the vocabulary, the model tries to break the word into the largest possible subwords in the vocabulary, and as a last resort will decompose the word into individual characters. Subwords which are not at the front are preceded by '##'. After splitting a word into subwords, the model holds the contextual meaning of the original word.

After breaking the text into tokens, *BERT* converts the sentence from a list of strings to a list of vocabulary indices as shown in Table 2.1. Here, the word 'bank' has same index ID with two different meaning. We have assumed that no bank (financial institution) is situated by the river. If it happens then, the model will be confused.

To distinguish between two consecutive sentences, *BERT* uses $1s$ and $0s$ as segmentation IDs. For example, if we have two successive sentences: $sentence_A$ and $sentence_B$, *BERT* will replace all the tokens of $sentence_A$ with $0s$ and all the tokens of $sentence_B$ with $1s$, shown as follows:

[0, 0, 0, 0, 0, 1, 1, 1, 1]

[$sentence_A$, $sentence_B$]

Table 2.1: Converting tokens into IDs using BERT

Tokens	**Indexed Tokens**
[*CLS*]	101
i	1,045
deposit	12,816
cash	5,356
in	1,999
the	1,996
bank	2,924
and	1,998
then	2,059
walk	3,328
by	2,011
the	1,996
bank	2,924
of	1,997
the	1,996
river	2,314
.	1,012
[*SEP*]	102

2.6.2 Word and Sentence Vectors

We can use the PyTorch interface to convert the data into a torch tensor where a torch tensor is a multi-dimensional matrix containing elements of a single data type. Then we can generate token embeddings using the torch tensor. We can create word vectors by summing the last four hidden layers for each token in the token embeddings. By averaging each token from the second last hidden layer, we can generate sentence vectors. These vectors are contextually dependent. For example, if we consider the input text shown in Table 2.1, we will have the word 'bank' with two different meanings. We load 'bert-base-uncased' tokenizer, where the model has 12 layers and 768 hidden units. Then the hidden state of the model will have four dimensions as follows:

- Number of layers: 13 (Initial embeddings + 12 BERT layers)
- Number of batches: 1 (In example, we have a single sentence)

- Number of tokens: 18 (Example sentence contains 18 tokens)
- Number of hidden units: 768

The shape of the word vector for the above example is 18×768. The first four vector-values for each instance of 'bank'are:

- **bank vault:** tensor([2.3105, -0.7636, -1.9906, 1.1273])
- **river bank:** tensor([-2.8783, 0.3906, -0.1861, 0.8590])

In this way, *BERT* holds the contextual meaning of the words, even if a word has two different meanings in the sequence.

2.7 Automatic Text Summarization

According to Jones and Endres-Niggemeyer (1995), automatic text summarization has two significant parts: finding the essential content of a document and expressing the selected sentences in a compressed fashion. There are two fundamental types of summaries; extractive and abstractive. In extractive summaries, contents are selected from the original document, whereas in abstractive summaries, some contents are paraphrased (Mani, 2001). An example of extractive and abstractive summaries is shown in Figure 2.11.

2.7.1 Extractive Summarization

Several researchers have explored two different methods for extractive summarization: unsupervised learning (Erkan and Radev, 2004; Fattah and Ren, 2009; Fang et al., 2017; Wang et al., 2008; Dunlavy et al., 2007; Parveen et al., 2015) and supervised learning (Li et al., 2013; Cheng and Lapata, 2016; Li et al., 2009; Fattah and Ren, 2009; Svore et al., 2007; Ouyang et al., 2011; Nallapati et al., 2017). The supervised learning model works on a labeled dataset as training; on the other hand, the unsupervised learning model works on unlabeled data. Unsupervised learning is developed by applying some model. Examples

Source Text:

Tom and Jerry went to kitchen to fight for food. Suddenly, Jerry felt terrible and returned to the home.

Extractive Summary:

Tom and Jerry fight for food. Jerry returned to home.

Abstractive Summary:

Jerry turned back home after fighting with Tom.

Figure 2.11: Types of Summary

of such models include clustering (Alguliyev et al., 2019; Shetty and Kallimani, 2017), graph-based method (Mallick et al., 2019; Dutta et al., 2019), language model (Gupta et al., 2011) and complex network-based method (Tohalino and Amancio, 2018). In supervised learning, numerous features are considered for sentence extraction. Kupiec et al. (1995) have demonstrated five elements for supervised learning: length of sentence, position in the paragraph, word frequency, uppercase words, and structure of phrase. Recently, Mao et al. (2019) proposed an approach by combining unsupervised and supervised learning to generate extractive summaries. The authors calculated scores of the sentences by using a graph-based unsupervised learning model. They considered the sentences as nodes and the relationships between sentences as edges. They calculated the scores of each node by using a graph solving algorithm and used the calculated scores as a feature along with other features to generate the summary using a supervised learning approach shown in Figure 2.12.

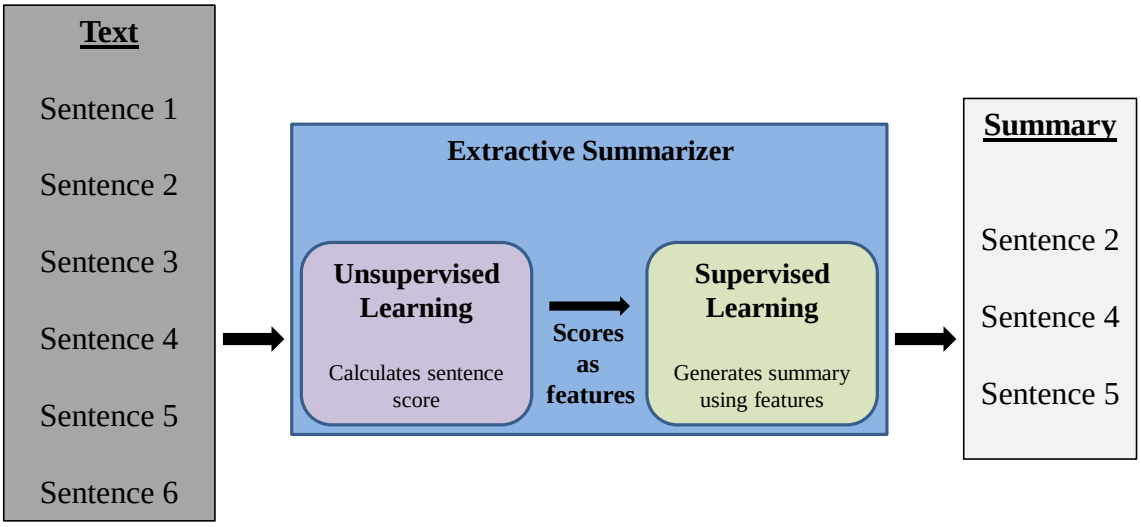


Figure 2.12: Extractive Summarization: Combining supervised and unsupervised learning

2.7.2 Abstractive Summarization

With the advancement of the neural network, modern approaches of text summarization have focused on abstractive summarization by using an encoder-decoder architecture (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017; Tan et al., 2017; Narayan et al., 2018). Rush et al. (2015) have used GigaWord and DUC corpus for their experiments. The work of Rush et al. (2015) has been extended by Chopra et al. (2016) to improve the performance of the two datasets. With the similar RNN encoder-decoder model, Hu et al. (2015) introduced a dataset for Chinese text summarization. To solve the problem of recurring words in encoder-decoder models, Chen et al. (2016) have given an attention model to minimize the repetition of the same words and phrases. To generate headlines from news articles, Lopyrev (2015) used a long short-term memory (*LSTM*) encoder-decoder model, which is shown in Figure 2.13. Recently, Song et al. (2019) incorporated the deep learning technique on the *LSTM* encoder-decoder model for abstractive summarization where they extracted phrases from the source sentences and then used deep learning. Deep learning is an artificial intelligence function where the artificial neural networks and the algorithms mimic the workings of the human brain. Their *LSTM* model is shown in Figure 2.14.

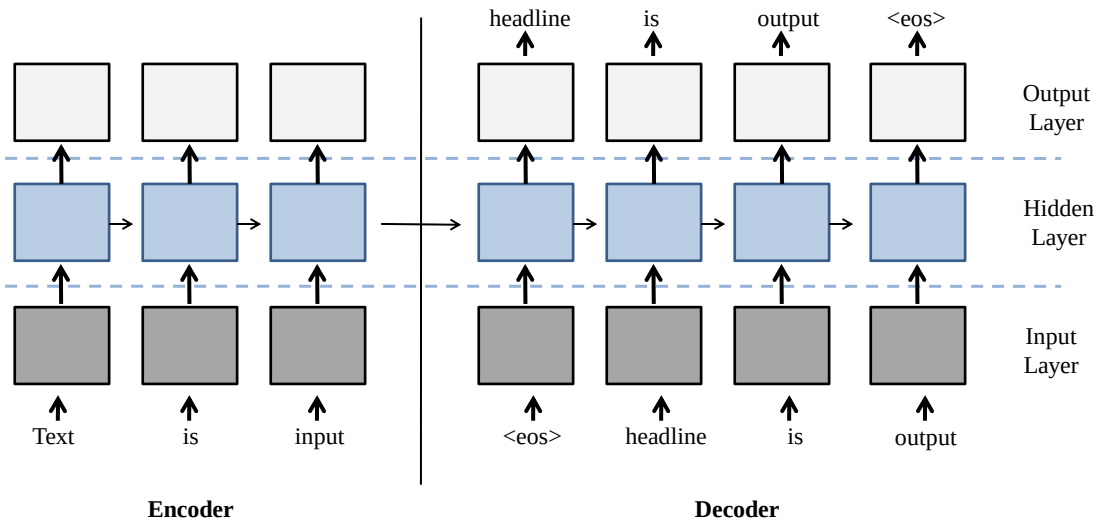


Figure 2.13: Encoder-Decoder Neural Network Architecture

2.8 Query Focused Summarization

In query focused summarization (*QFS*), documents are summarized according to the given query. For query focused extractive summarization, contents from the documents are selected according to the query relevance. To perform query focused abstractive summarization, researchers followed two steps:

- they performed the query focused extractive summarization
- then paraphrased the extracted summaries in the second step

Few works are proposed on query focused summarization using different models (Nema et al., 2017; Hasselqvist et al., 2017; Baumel et al., 2018; Lierde and Chow, 2019; Egonmwan et al., 2019).

2.8.1 Diversity Driven Attention Model

The diversity driven attention model was proposed by Nema et al. (2017) with two key additions to the encode-attend-decode model. The authors considered the attention model

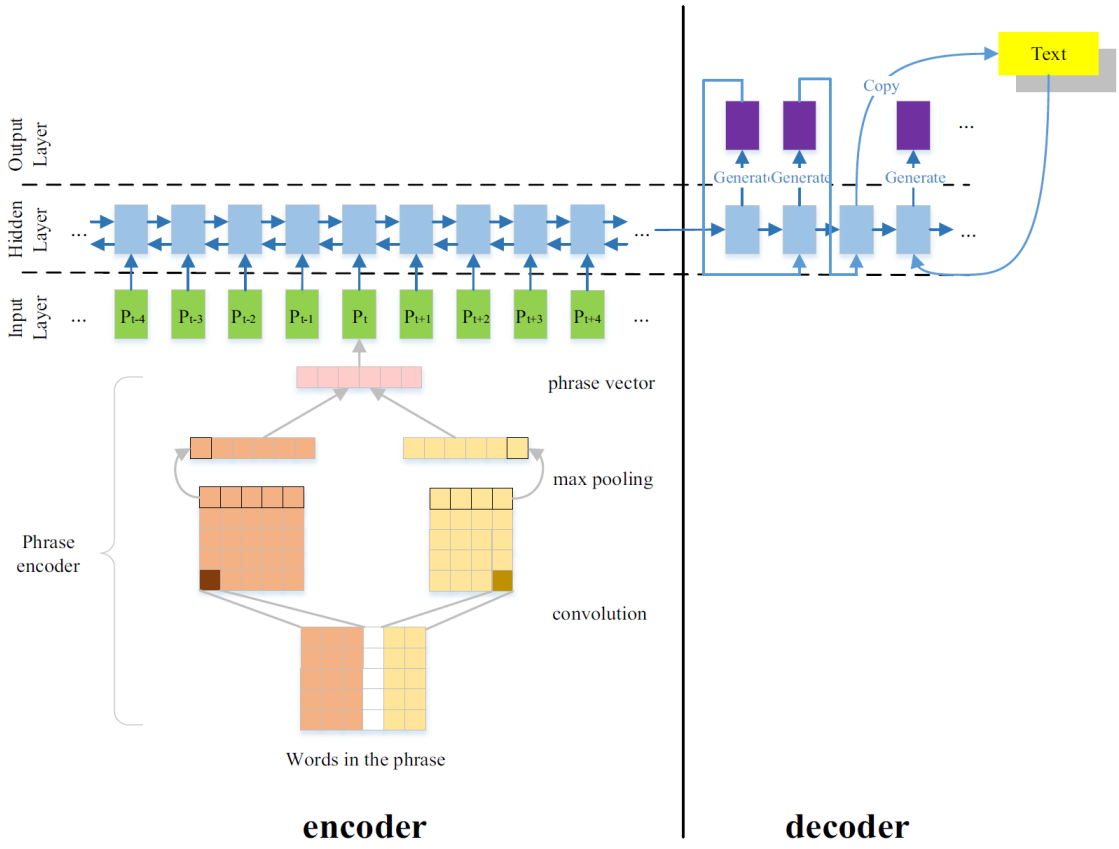


Figure 2.14: Semantic units based LSTM model (Song et al., 2019)

for both queries and documents. Their model succeeded in solving the problem of repeating phrases in summaries.

2.8.2 Sequence-to-Sequence Model

Hasselqvist et al. (2017) proposed a pointer-generator model for query focused abstractive summarization. The authors incorporated an attention and pointer generation mechanism on a sequence-to-sequence model. Baumel et al. (2018) incorporated query relevance in a pretrained abstractive model. In their approach, only the query-related sentences from multiple documents are passed to the abstractive model.

2.8.3 Hypergraph-based Summarization Model

To minimize the word repetition by selecting the essential contents from the document, Lierde and Chow (2019) have proposed a hypergraph-based summarization model. In their approach, they considered sentences as nodes and the relations between sentences as hyperedges.

2.8.4 Cross-Task Knowledge Transfer

Recently, Egonmwan et al. (2019) have proposed a query focused approach where after extracting the query-related sentences, the authors have used the off-the-shelf Machine Translation system to paraphrase the summary.

2.9 ROUGE: Performance Evaluation Tool

Lin (2004) developed a tool named ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to automatically determine the quality of a summary by comparing it to reference summaries created by humans. The evaluation between the computer-generated summary and the reference summaries are done by counting the number of overlapping units such as n-gram, word sequences, and word pairs. The author introduced four ROUGE measures: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S.

2.9.1 ROUGE-N

ROUGE-N is the n-gram recall measure between candidate summary and a set of reference summaries. For example, ROUGE-1, ROUGE-2, and ROUGE-3 measure unigram, bigram, and trigram overlap, respectively. ROUGE-N can be computed as follows:

$$ROUGE{-}N(recall) = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{2.8}$$

$$ROUGE{-}N(precision) = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{SystemSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{2.9}$$

Here, n is the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. The recall and precision, both measures are essential to evaluate a generated summary. So, the F-Score is better to be reported to calculate the accuracy as follows:

$$ROUGE{-}N(F{-}measure) = \frac{(1+\beta^2) \times ROUGE{-}N(recall) \times ROUGE{-}N(precision)}{ROUGE{-}N(recall) + \{\beta^2 \times ROUGE{-}N(precision)\}} \tag{2.10}$$

Here, $\beta = ROUGE{-}N(precision)/ROUGE{-}N(recall)$.

2.9.2 ROUGE-L (Longest Common Subsequence)

Given two sequences A and B, the longest common subsequence (LCS) of A and B is a common subsequence with maximum length. A Sentence-Level LCS can be considered as a subproblem to evaluate a summary. If we have a longer LCS of two sentences from two different summaries, then those summaries will be similar. For example, if A is a reference summary sentence with m words, and B is a candidate summary sentence with n words, we can calculate the ROUGE-L as follows:

$$ROUGE{-}L(recall) = \frac{LCS(A,B)}{m} \tag{2.11}$$

$$ROUGE{-}L(precision) = \frac{LCS(A,B)}{n} \tag{2.12}$$

$$ROUGE{-}L(F{-}measure) = \frac{(1+\beta^2) \times ROUGE{-}L(recall) \times ROUGE{-}L(precision)}{ROUGE{-}L(recall) + \{\beta^2 \times ROUGE{-}L(precision)\}} \tag{2.13}$$

Here, $LCS(A,B)$ is the length of the longest common subsequence of A and B, and $\beta = ROUGE{-}L(precision)/ROUGE{-}L(recall)$.

2.9.3 ROUGE-W (Weighted Longest Common Subsequence)

Basic LCS has a problem for which weighted LCS can be the right choice for the evaluation. For example, given a reference sequence A and two candidate sequences B and C as follows:

$$A\text{: } [\underline{word_1}, \underline{word_2}, \underline{word_3}, \underline{word_4}, word_5, word_6, word_7]$$

$$B\text{: } [\underline{word_1}, \underline{word_2}, \underline{word_3}, \underline{word_4}, word_8, word_9, word_{10}]$$

$$C\text{: } [\underline{word_1}, word_8, \underline{word_2}, word_{10}, \underline{word_3}, word_9, \underline{word_4}]$$

Here, B and C have the same ROUGE-L score. However, in this case, B should be the better choice than C because B has consecutive matches. ROUGE-W favors strings with consecutive matches. It can be computed efficiently using dynamic programming.

2.9.4 ROUGE-S

ROUGE-S uses skip-bigram, which allows arbitrary gaps in the sentence order. Skip-bigram cooccurrence statistics measure the overlap of skip-bigrams between a candidate translation and a set of reference translations. For example, if A is a reference translation with length m, and B is a candidate translation with length n, we can calculate the ROUGE-S as follows:

$$ROUGE{-}S(recall) = \frac{SKIP2(A,B)}{C(m,2)} \tag{2.14}$$

$$ROUGE{-}S(precision) = \frac{SKIP2(A,B)}{C(n,2)} \tag{2.15}$$

$$ROUGE{-}S(F{-}measure) = \frac{(1+\beta^2) \times ROUGE{-}S(recall) \times ROUGE{-}S(precision)}{ROUGE{-}S(recall) + \{\beta^2 \times ROUGE{-}S(precision)\}} \tag{2.16}$$

Here, $SKIP2(A,B)$ is the number of skip-bigram matches between A and B, β controls the relative importance of $ROUGE{-}S(recall)$ and $ROUGE{-}S(precision)$. C is a combination function that returns all possible bigram combinations of a translation.

2.10 Summary

In this chapter, we have described the background of our thesis, including the related works on automatic text summarization. Moreover, we have discussed the underlying architecture of the artificial neural network, which helps us to describe different summarization models. We have introduced the transformer model and then elaborated on the *BERT* model in this chapter. We have also discussed the *BERT* word embedding technique to visualize the input-output format of the *BERT* model. Finally, we have described the summary evaluation metrics, which is very important to measure our contributions by evaluating the summaries generated by our approaches.

Chapter 3

Generating Query Set

3.1 Introduction

In query focused summarization (*QFS*), the text documents are summarized according to the given queries. Among the two types of summarizations, query focused extractive summarization (*QFES*) has recently been trained and evaluated by using DUC[3] and SQuAD[4] (Roitman et al., 2020; Egonmwan et al., 2019). Few works have been done on query focused abstractive summarization (*QFAS*), and there is no suitable dataset which can be used to train and evaluate the *QFAS* models. We have generated and incorporated queries on two publicly available datasets to create an appropriate dataset for the *QFAS* models. Our selected datasets are CNN/Daily Mail[5] (Hermann et al., 2015) and Newsroom[6] (Grusky et al., 2018), which were used the most on abstractive summarization.

3.2 Datasets

We have processed the non-anonymized version of CNN/Daily Mail and Newsroom datasets which are available online as raw data containing the documents and summaries. As each dataset is stored in a different format, we have to fetch and process them into a common format.

[3] https://duc.nist.gov/data.html
[4] https://rajpurkar.github.io/SQuAD-explorer/
[5] https://cs.nyu.edu/~kcho/DMQA/
[6] http://lil.nlp.cornell.edu/newsroom/

3.2.1 CNN/Daily Mail

These datasets comprise news articles with related highlights where we have taken highlights as our summaries. In previous works, these datasets were mostly used for question answering. In our work, we have collected the stories as our text documents and the highlights as the corresponding summaries. In this way, the summaries contain more than one sentence, making the datasets more useful for the pretrained language models.

3.2.2 Newsroom

Grusky et al. (2018) developed the Newsroom dataset from 38 major news publications. The authors borrowed words and phrases from articles at varying rates to generate summaries by combining the abstractive and extractive approaches. In Newsroom, there are three types of datasets: *Abstractive*, *Extractive*, and *Mixed*. In the *Mixed* dataset, the authors extracted the sentences with some phrases from the articles to generate the summaries. For the last two years, the *Mixed* dataset has been used to generate abstractive summaries (Shi et al., 2019; Subramanian et al., 2019; Egonmwan and Chali, 2019b), where the authors preprocessed the dataset before creating the abstractive summaries. We are also using the *Abstractive* dataset of Newsroom to compare the results with the works of Rush et al. (2015) and Liu and Lapata (2019). We have taken those documents which have the length of summaries of at least two sentences.

3.3 Query Generation Architecture

Generally, the summarizer can use the query to make the concise text. The query can be a set of keywords instead of an expression as an interrogative sentence. For example, in the information retrieval system, if we search for the 'influenza virus' then the system will return the information related to the 'virus' whose name is 'influenza'. In the summarization approach, we can also consider the query as a set of keywords.

When we search in a text with our given query, we expect the presence of those query

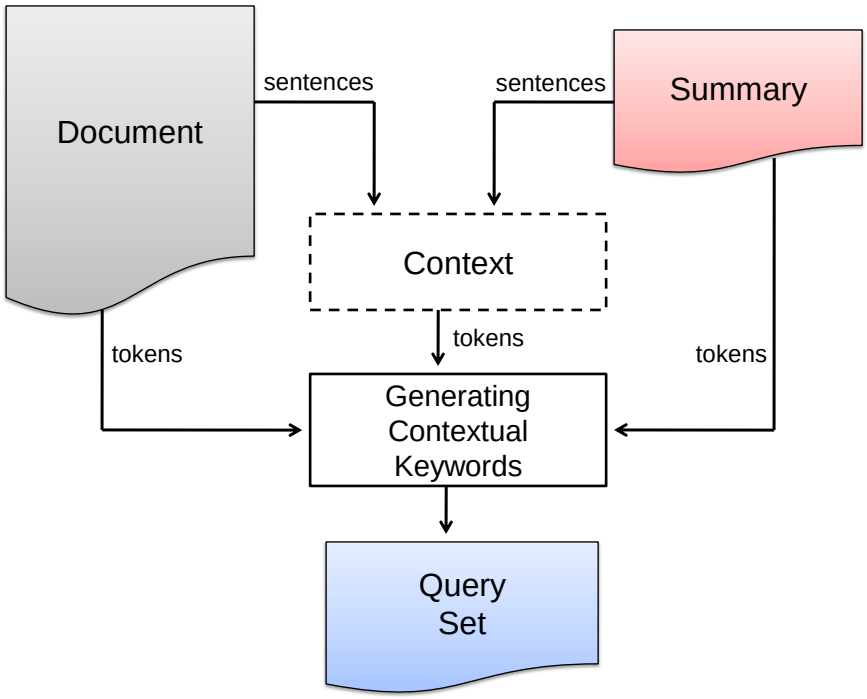


Figure 3.1: Query Generation Architecture

keywords in our search results. In query focused summarization, both the generated summary as well as the source document should contain the query keywords. For example, if we want to summarize a document that contains the weather report of Alberta, and someone wants to know about Lethbridge's weather as a summary by providing a query ('weather' 'lethbridge'), then the main document should contain these two keywords: 'weather' and 'lethbridge'. Otherwise, we can assume that the source document has no information regarding Lethbridge's weather report, and both document and query will be considered invalid. Similarly, in the summary, the presence of these two keywords will confirm that the summary is query focused. The query holds the context of the summary, where the query keywords should be present in the source document. For this reason, we have considered the common words between document and summary as keywords of the query. Our query generation architecture is shown in Figure 3.1. To find important words in a document, we have used *TF-IDF*.

***TF*:** Term Frequency (*TF*) is the word's frequency in a document.

***IDF*:** Inverse Document Frequency (*IDF*) decreases the weight of term which is frequent but not meaningful.

***TF-IDF*:** *TF-IDF* is a numerical statistic that is intended to reflect how important a word is to a text.

As the query should hold the keywords which are in both the document and summary, the top common words from the document and summary are listed as keywords. First, we have generated the sparse matrices of the document and summary by using *CountVectorizer* of the *sklearn* library (Pedregosa et al., 2011) to convert the text into a matrix of token counts. Then we have computed the *TF-IDF* of the document and summary by using *TfidfTransformer* of the *sklearn* library. Finally, we have extracted the top five common keywords from the *TF-IDF* vectors of document and summary. Our query generation process is shown in Algorithm 1.

Algorithm 1: Generate a query from a text document and summary

generatingQuery (D, S)

Data: takes a text document, D and its corresponding summary, S as input

Result: returns D, S and the generated query Q

source = textProcessing (D);

$target$ = textProcessing (S);

$wordCountVector_D$ = CountVectorizer.fit_transform($source$);

$wordCountVector_S$ = CountVectorizer.fit_transform($target$);

$tfidfVector_D$ = TfidfTransformer.fit_transform($wordCountVector_D$);

$tfidfVector_S$ = TfidfTransformer.fit_transform($wordCountVector_S$);

$Q = [\,]$;

$Q =$ selectCommonKeywords($tfidfVector_D, tfidfVector_S, 5$);

return Q;

3.3.1 Process Text Document and Summary

We need to process the raw texts to find the common words between the given document and summary. We have used the CoreNLP toolkit (Manning et al., 2014) for splitting the sentences from each document and followed the approach of See et al. (2017) to preprocess the datasets. In the 'textProcessing' function, we have prepared the text document and the corresponding summary. This function takes the sequence of sentences as input. Then it removes all the special characters and the stop words from the text. Finally, the function lemmatizes the text and returns all the necessary alphanumeric words as a list of tokens.

3.3.2 Generating Contextual Keywords

Generally, the words related to the context of a document are important. By using the *CountVectorizer* of the *sklearn* library (Pedregosa et al., 2011), we have converted the text into a matrix of token counts. Then we have generated two *TF-IDF* vectors to measure the importance of words in the texts for the document and summary.

3.3.3 Finalizing Keywords as Query

We have selected the five common tokens between the *TF-IDF* vectors of document and summary whose *TF-IDF* values are higher. The 'selectCommonKeywords' function takes two *TF-IDF* vectors of the document and summary, and the number of keywords as its input. Then the function creates a list of common words between two *TF-IDF* vectors. Each word in the list holds a weight, which is calculated by adding the *TF-IDF* values from two vectors. Finally, the function returns a list of keywords as the query whose weights are higher in the list.

3.4 Prepared Datasets

We have prepared the datasets for the query focused abstractive summarization by incorporating the generated queries into both datasets CNN/Daily Mail and Newsroom. A sample of both datasets is shown in Figure 3.2.

CNN/Daily Mail

Source Document: (cnn) a mom furious at her son for apparently taking part in the baltimore riots has become a sensation online . in video captured by cnn affiliate wmar , the woman is seen pulling her masked son away from a crowd , smacking him in the head repeatedly , and screaming at him . wmar reports that the woman saw her son on television throwing rocks at police . but police commissioner anthony batts thanked her in remarks to the media . " and if you saw in one scene you had one mother who grabbed their child who had a hood on his head and she started smacking him on the head because she was so embarrassed , " he said monday .

Query: "police", "son", "saw", "cnn", "mom"

Summary: the mom saw her son on tv throwing rocks at police , cnn affiliate reports . police praise her actions

Newsroom

Source Document: the kidz bop kids are a live - tour extension of the kidz bop album series , which updates the latest pop hits into kid - friendly versions . " hi ! my name is cassie . i ' m 8 years old , and i ' m going to sing ' anaconda ' by nicki minaj . " that does not happen at a kidz bop kids show , of course , but you ' d be forgiven for expecting such a jaw - dropping scenario . . . (length of the source document is 833 words)

Query: "kidz", "bop", "album", "series" , "come"

Summary: nearly 15 years on , the massively successful kidz bop album franchise proves it wasn ' t just a trend . the series ' traveling ensemble , the kidz bop kids , come to wilbur theatre on june 27

Figure 3.2: Sample Datasets with Generated-Query

3.5 Summary

In this chapter, we have described the process to generate the query for a given document and the corresponding summary. We have described the datasets collection strategies and the necessity to incorporate queries in the datasets. By using our developed algorithm, we have prepared two publicly available datasets, CNN/Daily Mail, and Newsroom for the query focused summarization models.

Chapter 4

Abstractive Summarization using Pretrained Language Model (BERTSUM)

4.1 Introduction

In this thesis, we have developed two approaches for query focused abstractive summarization using the *BERTSUM* model which are described in Chapters 5 and 6. In this chapter, we have initially implemented an abstractive summarization approach using the *BERTSUM* model, where the query was not considered. To apply the approach, we have followed the work of Liu and Lapata (2019). This implementation has helped us to compare the results with and without considering the query for the abstractive summarization approaches.

4.2 Overview of the Model

We have reconstructed the *BERT* encoder and decoder to use the model for the summarization task. The basic *BERT* model gives the output as tokens instead of sentences; however, we have provided the output as a sequence of sentences for summarization. Following the work of Liu and Lapata (2019), we have considered the sentences individually by incorporating the $[CLS]$ token at the start of each sentence. To identify multiple sentences in a document, we have used an interval-segment, where we have assigned sentence embeddings E_A and E_B depending on the positions of the sentences. In our approach, we

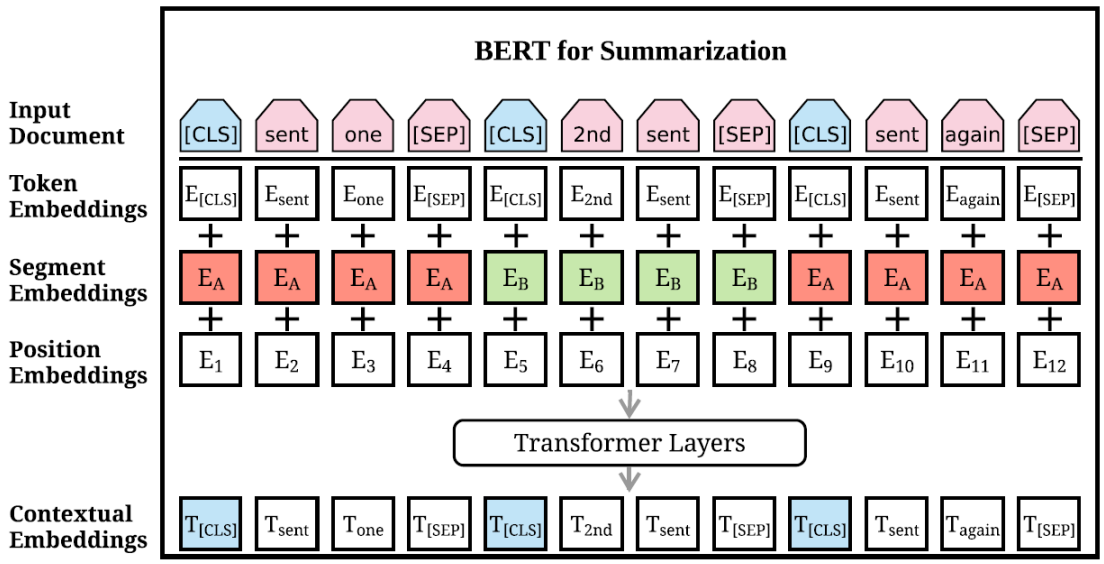


Figure 4.1: Architecture of the BERTSUM model for Summarization (Liu and Lapata, 2019).

have used more position embeddings that are initialized randomly. Finally, the summation of three embeddings (*Token*, *Segment*, and *Position*) of the input document is passed to the transformer. The modified *BERT* model, namely *BERTSUM* (Liu and Lapata, 2019) that we have followed to implement the summarization task is shown in Figure 4.1.

We have followed the standard encoder-decoder framework (See et al., 2017), where the encoder is pretrained and the decoder is initialized randomly with the 6-layered transformer. As the encoder is pretrained, and the decoder is trained from scratch, one of them might overfit while the other one underfits the data. We have used the Adam optimizers (Kingma and Ba, 2014), $\beta_1 = 0.9$ for the encoder, and $\beta_2 = 0.999$ for the decoder to make the fine-tuning stable and also have used the learning rates for encoder and decoder as in the following equations:

$$lr_{\mathcal{E}} = \tilde{lr}_{\mathcal{E}}.min(step^{-0.5}, step.warmup_{\mathcal{E}}^{-1.5}) \tag{4.1}$$

$$lr_{\mathcal{D}} = \tilde{lr}_{\mathcal{D}}.min(step^{-0.5}, step.warmup_{\mathcal{D}}^{-1.5}) \tag{4.2}$$

where, for the encoder, $warmup_{\mathcal{E}} = 20,000$ and $\tilde{lr}_{\mathcal{E}} = 2e^{-3}$, and $warmup_{\mathcal{D}} = 10,000$ and $\tilde{lr}_{\mathcal{D}} = 0.1$ for the decoder. In this way, the pretrained encoder is fine-tuned with a lower learning rate.

4.3 Experimental Setup

In this section, we have presented our experimental setup for assessing the performance of our implemented pretrained abstractive summarization model. Here, we have explained the dataset preparation and evaluation metrics to determine the performance of the model. We have presented some baseline systems to compare our results with them.

4.3.1 Dataset

We have evaluated the model on two publicly available datasets, CNN/Daily Mail (Hermann et al., 2015), and Newsroom (Grusky et al., 2018). We have split the sentences of the documents and summaries using the CoreNLP toolkit (Manning et al., 2014) and then preprocessed both datasets following the work of See et al. (2017).

CNN/Daily Mail

We have considered the stories as the documents and the related highlights as the summaries from the CNN/Daily Mail dataset. A brief description on CNN/Daily Mail has been provided in Section 3.2. We have followed the approach of Hermann et al. (2015) to split the datasets into the training, testing, and validation sets. We have split the CNN dataset as 90,266 documents for the training, 1,093 documents for the testing, and 1,220 documents for the validation. The Daily Mail dataset has been split as 196,961 documents for the training, 10,397 documents for the testing, and 12,148 documents for the validation.

Newsroom

In Newsroom, there are three types of datasets: *Abstractive*, *Extractive*, and *Mixed*. In the *Extractive* dataset, meaningful sentences are selected from the document to construct

the summary. Human-generated sentences are considered as a summary of the document in the *Abstractive* dataset. In the *Mixed* dataset, the summary is generated by combining the extractive and abstractive approaches. In this chapter, we have used the *BERTSUM* model for the abstractive summarization. So, we have used the *Abstractive* dataset and the *Mixed* dataset of the Newsroom for our experiments. We have taken those documents which have the length of summaries of at least two sentences and followed the approach of Grusky et al. (2018) to split the datasets into the training, testing, and validation sets. We have split the *Abstractive* dataset as $59,087$ documents for the training, $6,371$ documents for the testing, and $6,372$ documents for the validation. The *Mixed* dataset has been split as $48,407$ documents for the training, $5,240$ documents for the testing, and $5,189$ documents for the validation.

4.3.2 Implementation Details

We have trained the model for $200,000$ steps on a TITAN GPU (GTX Machine). We have used PyTorch, OpenNMT (Klein et al., 2017), and incorporated a pretrained language model, 'bert-base-uncased' of the *BERT* model (Devlin et al., 2019) on the encoder and used the dropout probability 0.1 and the label-smoothing factor 0.1 (Szegedy et al., 2016). For the decoder, we have taken 768 hidden units with the hidden size for feed-forward layers as $2,048$. In the decoding phase, we have used a beam size 5, and for the length penalty, we have tuned α between 0.6 and 1.0 (Wu et al., 2016).

4.3.3 Evaluation Metric

We have evaluated our experiments using ROUGE-1 (unigram), ROUGE-2 (bigrams), and ROUGE-L (Longest Common Subsequence) (Lin, 2004), which calculate the word-overlapping between the reference and the system summaries. We have described about the ROUGE metrics in Section 2.9.

4.3.4 Baseline Systems

We have experimented with three different datasets: CNN/Daily Mail, Abstractive Newsroom, and Mixed Newsroom. To compare our experimental results with the baseline and state-of-the-art results, we have collected the results from Liu and Lapata (2019) and Egonmwan and Chali (2019b) for the CNN/Daily Mail datasets and the results from Grusky et al. (2018) and Egonmwan and Chali (2019b) for the Newsroom datasets.

Table 4.1: ROUGE-F1 (%) scores (with 95% confidence interval) of the *BERTSUM* model implementation and various abstractive models on the CNN/Daily Mail test set. (*: taken from Liu and Lapata (2019). †: taken from original paper.)

Model	**R1**	**R2**	**RL**
PTGEN * (See et al., 2017)	36.44	15.66	33.42
PTGEN+COV * (See et al., 2017)	39.53	17.28	36.38
DRM * (Paulus et al., 2017)	39.87	15.82	36.90
BOTTOMUP * (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA * (Çelikyilmaz et al., 2018)	41.69	**19.47**	37.92
TRANS-EXT+FILTER+ABS † (Egonmwan and Chali, 2019b)	**41.89**	18.90	**38.92**
TRANSFORMERABS * (Liu and Lapata, 2019)	40.21	17.76	37.09
BERTSUM (OUR IMPLEMENTATION)	40.78	18.64	37.95

4.4 Result and Discussion

Tables 4.1, 4.2, and 4.3 summarize the results of the CNN/Daily Mail dataset, the Abstractive dataset of Newsroom, and the Mixed dataset of Newsroom, respectively. Tables 4.1 and 4.3 show that our implementation on both of these datasets could not perform better than most of the recent baseline and state-of-the-art results. However, Table 4.2 shows that our implementation performed better than the work of Rush et al. (2015), which was not a recent work. Most of the latest works are considering different techniques before generating the abstractive summaries. In these experiments, we have just produced the abstractive summaries using the *BERTSUM* model and did not incorporate any additional technique before producing summaries. From these experiments, we hypothesized that without considering the query, *BERTSUM* could not provide better results. We have proposed two

approaches by incorporating the query relevance in the *BERTSUM* model to obtain better results, as discussed in Chapters 5 and 6.

4.5 Summary

In this chapter, we used the *BERTSUM* model for abstractive summarization. Though the model did not show better results than the baseline and state-of-the-art results, this implementation helped us to know about the model architecture and the behavior on the datasets. Comparing the results of the other models, we hypothesized that incorporating query relevance in the *BERTSUM* model can help to generate state-of-the-art summaries. In Chapters 5 and 6, we have described two of our approaches where we have incorporated the query relevance in the *BERTSUM* model.

Table 4.2: ROUGE-F1 (%) scores (with 95% confidence interval) of the *BERTSUM* model implementation and various abstractive models on the Newsroom (Abstractive) test set. (*: taken from Grusky et al. (2018).)

Model	**R1**	**R2**	**RL**
SEQ2SEQ + ATTENTION * (Rush et al., 2015)	5.99	0.37	5.41
BERTSUM (OUR IMPLEMENTATION)	**15.05**	**1.90**	**13.46**

Table 4.3: ROUGE-F1 (%) scores (with 95% confidence interval) of the *BERTSUM* model implementation and various abstractive models on the Newsroom (Mixed) test set. (*: taken from Grusky et al. (2018). †: taken from original paper.)

Model	**R1**	**R2**	**RL**
MODIFIED P-G * (Shi et al., 2019)	**39.91**	**28.38**	**36.87**
TLM * (Subramanian et al., 2019)	33.24	20.01	29.21
POINTER * (See et al., 2017)	26.04	13.24	22.45
TRANS-EXT+FILTER+ABS † (Egonmwan and Chali, 2019b)	35.74	16.52	30.17
BERTSUM (OUR IMPLEMENTATION)	32.09	15.21	28.88

Chapter 5

Query Focused Abstractive (QAbs) Approach

5.1 Introduction

In this chapter, we have designed one of our unique approaches to generate abstractive summaries by incorporating the query relevance in the *BERTSUM* model. Following the summarization technique of Liu and Lapata (2019), we have fine-tuned the *BERTSUM* model with our prepared query friendly datasets: CNN/Daily Mail (Hermann et al., 2015) and Newsroom (Grusky et al., 2018).

5.2 Overview of the Model

BERTSUM is a modified model of a pretrained language model, namely *BERT* that has been trained on a corpus of 3.3 Billion words and can perform the 'next sentence prediction' task. Moreover, by considering each sentence as a separate sequence, *BERTSUM* can take all the sentences of a document as input. But the sequence of the sentences is vital to fine-tune the model. The next sentence prediction for the summary should be query relevant in query focused abstractive summarization. All the query-related sentences should be at the beginning of a document so that *BERTSUM* can be fine-tuned with query-related sentences one by one. To get the document's total context, the sentences which are not associated with the query can also be considered after fine-tuning the model with the query-related sentences. The document's total context might help the the *BERTSUM* model to predict query focused words that will be more suitable for the summary.

Our summarization framework has three parts. In the first part, we have made queries according to the context of the documents and summaries of both publicly available datasets, CNN/Daily Mail, and Newsroom. Next, in the second part, we have sorted the sentences of the documents according to the relevance of the queries. The main reason for sorting the documents is to place the query-related sentences at the top of the document. In the third part, we have fine-tuned the *BERTSUM* model with all the query-related sentences one by one from the sorted documents and then considered the remaining sentences for fine-tuning the model to get the overall context of the document. In this approach (*QAbs*), we have selected all the sentences from the sorted documents for fine-tuning the model.

5.2.1 Generating Query Set

We have modified two datasets: CNN/Daily Mail and Newsroom by generating and incorporating the queries in both datasets. Our query generation approach has been described in Chapter 3.

5.2.2 Sentence Ordering

We have sorted the sentences of a document according to the relevance of the generated query with each sentence. Let the document be D with a sequence of sentences $\{S_1, S_2, ..., S_n\}$ and the generated query be Q with a set of keywords $\{q_1, q_2, ..., q_m\}$, where n is the number of sentences in the document and m is the number of keywords in the query. We have ordered the sentences by calculating the relevance with the keywords of the query to get sorted document, $D_{SORT} = \{..., S_i, S_j, ...\}$, where, $1 \leq i, j \leq n$; $i \neq j$; and $similarity(Q, S_i) \geq similarity(Q, S_j)$, i.e., S_i is more related with the query than S_j. The number of similar words between Q and S is divided by the length of Q to calculate the similarity between Q and S. Our sentence ordering algorithm is shown in Algorithm 2. Here, the function 'sortDescendingSimilarity' takes a document as an input, which contains a sequence of sentences and their similarity scores and returns a sequence of sorted sentences in descending order according to their similarity scores.

Algorithm 2: Sort a document according to its query relevance

sortingDocument (D, Q)

Data: takes a text document, D and a query, Q as input

Result: returns the sorted document, D_{SORT}

$doc = \{\}$;

foreach $S \in D$ **do**

$count = 0$;

foreach $word \in S$ **do**

if $word \in Q$ **then**

$count = count + 1$;

end

end

$similarity = count/len(Q)$;

doc.append (S, $similarity$);

end

$D_{SORT} = [\,]$;

D_{SORT} = sortDescendingSimilarity(doc);

return D_{SORT};

5.2.3 Fine-tune the Pretrained Language Model

Initially, we have split all the sentences into tokens and incorporated the *[CLS]* token at the beginning and *[SEP]* token at the end of each sentence of D_{SORT}. Each token is assigned three embeddings: token, segmentation, and position embeddings. Token embedding has been used to represent the meaning of each token, whereas segmentation embedding is used to identify each sentence separately. Finally, the position embedding has been used to determine the position of each token. We have summed three embeddings of input documents before passing to the transformer layers, as shown in Figure 5.1.

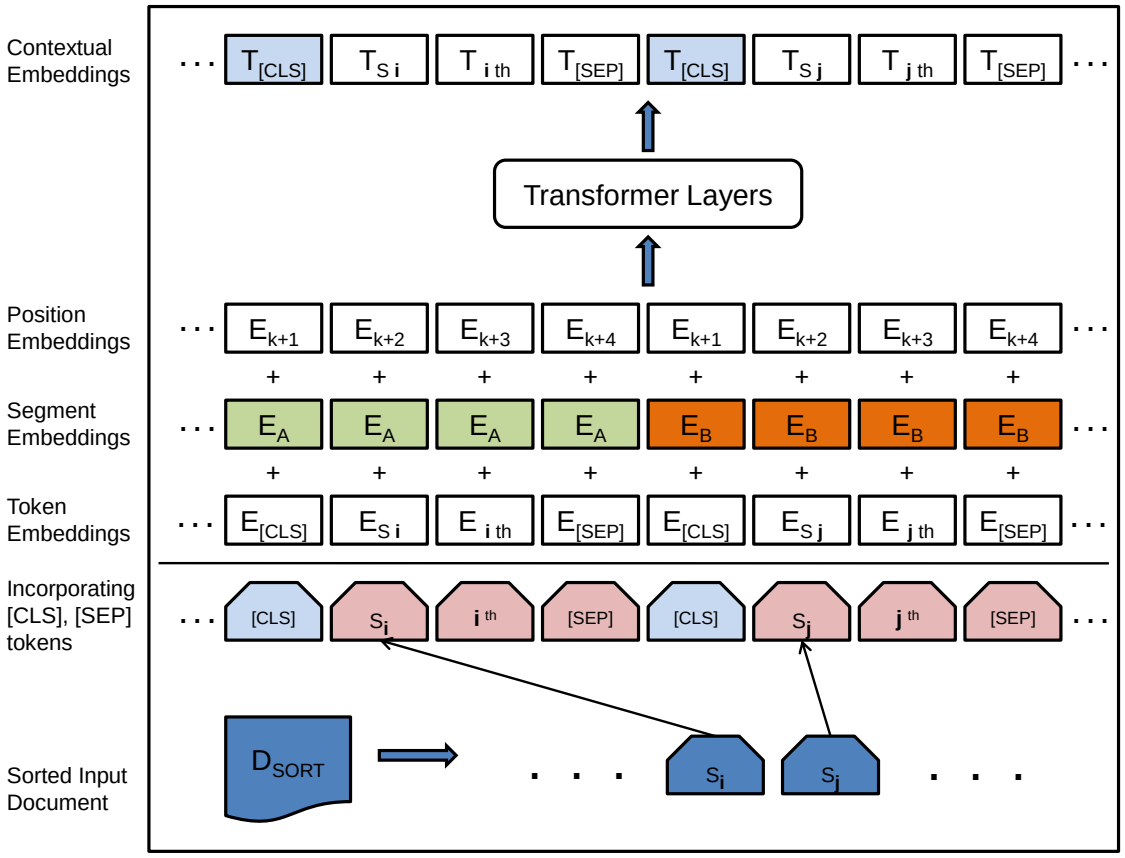


Figure 5.1: Architecture of improved BERTSUM model. Here, in position embedding, $k = \{4 \times (pos - 1)\}$, where pos is the position of a sentence in D_{SORT}

We have followed the same encoder-decoder framework, which we have used for implementing the *BERTSUM* model for abstractive summarization in Chapter 4. We have set the Adam optimizers (Kingma and Ba, 2014), $\beta_1 = 0.9$ for the encoder, and $\beta_2 = 0.999$ for the decoder to make our fine-tuning stable. The learning rates for the encoder and decoder are used as Equations 4.1 and 4.2, respectively.

5.3 Experimental Setup

To perform the fair comparison between our implemented approaches, we have followed the same experimental setup described in Section 4.3. The overall comparison takes place in Chapter 7.

5.3.1 Dataset

We have split the CNN/Daily Mail and Newsroom datasets into the training, testing, and validation sets as described in Section 4.3.1.

5.3.2 Implementation Details

We have run our experiments on a TITAN GPU (GTX Machine) and used PyTorch, OpenNMT (Klein et al., 2017), and the 'bert-base-uncased' version of *BERT* to implement our approach. We have set the dropout probability 0.1 and the label-smoothing factor 0.1 (Szegedy et al., 2016). In the decoding phase, 768 hidden units with the hidden size for feed-forward layers $2,048$ have been used where the beam size is 5, and the range of the length penalty, α has been considered in between 0.6 and 1.0 (Wu et al., 2016).

5.3.3 Evaluation Metric

We have used ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) to calculate the word-overlapping between the reference and the system summaries. A brief description of these evaluation metrics is given in Section 2.9.

5.3.4 Baseline Systems

We have compared our approach with the baseline and state-of-the-art systems which are collected from the works of Liu and Lapata (2019) and Egonmwan and Chali (2019b), regarding the CNN/-Daily Mail datasets, and from the works of Grusky et al. (2018) and Egonmwan and Chali (2019b), regarding the Newsroom datasets.

5.4 Result and Discussion

Tables 5.1, 5.2, and 5.3 summarize the results on the CNN/Daily Mail dataset, the *Abstractive* dataset of Newsroom, and the *Mixed* dataset of Newsroom, respectively. Tables 5.1 and 5.2 show that our approach on CNN/Daily Mail and Newsroom(*Abstractive*) test sets achieve better scores on ROUGE than the other models. For the *Mixed* dataset of

Newsroom, Table 5.3 shows that our approach performs better on ROUGE-1 and ROUGE-L. On ROUGE-2, the model has a drop of value against the current state-of-the-art model, and one of the possible reasons may the size of the dataset, which was small to fine-tune the *BERTSUM* model for generating summaries like a human.

Table 5.1: ROUGE-F1 (%) scores (with 95% confidence interval) of our *QAbs* approach and various abstractive models on the CNN/Daily Mail test set. (*: taken from Liu and Lapata (2019). †: taken from original paper.)

Model	**R1**	**R2**	**RL**
PTGEN * (See et al., 2017)	36.44	15.66	33.42
PTGEN+COV * (See et al., 2017)	39.53	17.28	36.38
DRM * (Paulus et al., 2017)	39.87	15.82	36.90
BOTTOMUP * (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA * (Çelikyilmaz et al., 2018)	41.69	19.47	37.92
TRANS-EXT+FILTER+ABS † (Egonmwan and Chali, 2019b)	41.89	18.90	38.92
TRANSFORMERABS * (Liu and Lapata, 2019)	40.21	17.76	37.09
QABS (OUR APPROACH)	**44.91**	**21.81**	**41.70**

Table 5.2: ROUGE-F1 (%) scores (with 95% confidence interval) of our *QAbs* approach and various abstractive models on the Newsroom (Abstractive) test set. (*: taken from Grusky et al. (2018).)

Model	**R1**	**R2**	**RL**
SEQ2SEQ + ATTENTION * (Rush et al., 2015)	5.99	0.37	5.41
QABS (OUR APPROACH)	**15.05**	**2.26**	**13.50**

5.5 Summary

In this chapter, we have implemented one of our novel approaches, *QAbs*, where we have incorporated the query relevance in the *BERTSUM* model. Our experimental results show that our model achieves new state-of-the-art results on ROUGE-1 and ROUGE-L. The approach shows better results on ROUGE-2 for the CNN/Daily Mail and the *Abstractive* dataset of Newsroom.

Table 5.3: ROUGE-F1 (%) scores (with 95% confidence interval) of our *QAbs* approach and various abstractive models on the Newsroom (Mixed) test set. (*: taken from Grusky et al. (2018). †: taken from original paper.)

Model	**R1**	**R2**	**RL**
MODIFIED P-G * (Shi et al., 2019)	39.91	**28.38**	36.87
TLM * (Subramanian et al., 2019)	33.24	20.01	29.21
POINTER * (See et al., 2017)	26.04	13.24	22.45
TRANS-EXT+FILTER+ABS † (Egonmwan and Chali, 2019b)	35.74	16.52	30.17
QABS (OUR APPROACH)	**40.67**	22.66	**36.92**

Chapter 6

Query Focused Extractive then Abstractive (QExAbs) Approach

6.1 Introduction

In this chapter, we have designed our second approach to generate abstractive summaries where queries have been incorporated in the *BERTSUM* model. In our Query Focused Extractive then Abstractive (*QExAbs*) approach, we did not consider all the sentences of the document for fine-tuning the *BERTSUM* model, which is the main difference between our two approaches. For generating abstractive summaries, the most common practice is to extract the salient sentences from the document and then paraphrase them. In our first approach, *QAbs*, we used the pretrained language model, where we considered all the sentences of a document for the fine-tuning. However, in this approach, *QExAbs*, we have only extracted the query-related sentences to fine-tune the *BERTSUM* model.

6.2 Overview of the Model

We can fine-tune the pretrained language model (*BERTSUM*) by our prepared datasets to generate abstractive summaries. For the fine-tuning, we only need to select the query-related sentences as the document's total context is not necessary to generate query focused summaries. Incorporating an additional step in the *QAbs* approach, we have designed the summarization framework of *QExAbs* with four steps: Generating Query Set, Sentence Ordering, Extracting Query Related Sentences, and Fine-tune the Pretrained Language Model.

6.2.1 Generating Query Set

Generating queries for the CNN/Daily Mail and Newsroom datasets to make them query friendly is one of our contributions. We have described the query generation process in Chapter 3.

6.2.2 Sentence Ordering

Ordering the sentences of a document according to the given query is our novel approach. We have sorted the sentences of the documents according to the queries before extracting the sentences from the document. In Section 5.2.2, we described our sentence ordering approach.

6.2.3 Extracting Query Related Sentences

We have extracted the query-related sentences in this phase. We first measured the similarity of each sentence with the keywords of the query to sort the sentences using Algorithm 2. We performed the sorting operation by calling the 'sortDescendingSimilarity' function. Besides, we called for another function named 'extractSimilarSentences' to return a sequence of sentences with similarity values greater than zero. The 'extractSimilarSentences' function also considers the sentences which have very low similarity values. The sentences which have a similarity value of zero are not the query-related sentences and hence were not considered for fine-tuning the *BERTSUM* model.

6.2.4 Fine-tune the Pretrained Language Model

After extracting the query-related sentences, we have split each sentence into tokens and incorporated the *[CLS]* token at the beginning and the *[SEP]* token at the end of each sentence. We have assigned token embedding, segmentation embedding and position embedding for each token. We have summed the three embeddings for each input sentence and then passed the result to the transformer layers, as shown in Figure 5.1. Similar to our *QAbs* approach, we have followed the same encoder-decoder framework described in Chapter 4.

We have set the Adam optimizers (Kingma and Ba, 2014), $\beta_1 = 0.9$ for the encoder, and $\beta_2 = 0.999$ for the decoder to make our fine-tuning stable. The learning rates for encoder and decoder are used as Equations 4.1 and 4.2, respectively.

6.3 Experimental Setup

We have followed the same experimental structure described in Section 4.3 to perform the fair comparison between our implemented approaches. The overall comparison takes place in Chapter 7.

6.3.1 Dataset

We have used the CNN/Daily Mail and Newsroom datasets for our experiment. We have split the CNN/Daily Mail and Newsroom datasets into the training, testing, and validation sets as described in Section 4.3.1.

6.3.2 Implementation Details

We have used PyTorch, OpenNMT (Klein et al., 2017), and the 'bert-base-uncased' version of *BERT* to implement our approach and run our experiments on a TITAN GPU (GTX Machine). We have tokenized the source and target texts using BERT's subword tokenizer. We have set the dropout probability 0.1 and the label-smoothing factor 0.1 (Szegedy et al., 2016). For decoding, we have used 768 hidden units with the hidden size for feed-forward layers $2,048$ and set the beam size 5. The range of the length penalty, α is set to between 0.6 and 1.0 (Wu et al., 2016).

6.3.3 Evaluation Metric

We have evaluated our approach using the ROUGE metrics (Lin, 2004) that calculate the word-overlap between the reference and the system summaries as described in Section 2.9.

6.3.4 Baseline Systems

We have collected the results of the baseline and state-of-the-art systems considering both CNN/Daily Mail and Newsroom datasets to compare our approach. Results are taken from the work of Liu and Lapata (2019) and Egonmwan and Chali (2019b) to compare using the CNN/-Daily Mail datasets. Some results are taken from Grusky et al. (2018) and Egonmwan and Chali (2019b) for comparison using the Newsroom datasets.

6.4 Result and Discussion

Our experimental results considering the CNN/Daily Mail dataset, the *Abstractive*, and the *Mixed* datasets of Newsroom are shown in Tables 6.1, 6.2, and 6.3, respectively. Results show that our approach achieves state-of-the-art results on ROUGE-1 and ROUGE-L. On ROUGE-2, our model performs better for the CNN/Daily Mail and the *Abstractive* dataset of Newsroom shown in Tables 6.1 and 6.2. But in Table 6.3, the model has a drop of value against the current state-of-the-art model on ROUGE-2, and one of the possible reasons may the size of the dataset, which was small to fine-tune the *BERTSUM* model for generating summaries like a human.

Table 6.1: ROUGE-F1 (%) scores (with 95% confidence interval) of our *QExAbs* approach and various abstractive models on the CNN/Daily Mail test set. (*: taken from Liu and Lapata (2019). †: taken from original paper.)

Model	**R1**	**R2**	**RL**
PTGEN * (See et al., 2017)	36.44	15.66	33.42
PTGEN+COV * (See et al., 2017)	39.53	17.28	36.38
DRM * (Paulus et al., 2017)	39.87	15.82	36.90
BOTTOMUP * (Gehrmann et al., 2018)	41.22	18.68	38.34
DCA * (Çelikyilmaz et al., 2018)	41.69	19.47	37.92
TRANS-EXT+FILTER+ABS † (Egonmwan and Chali, 2019b)	41.89	18.90	38.92
TRANSFORMERABS * (Liu and Lapata, 2019)	40.21	17.76	37.09
QEXABS (OUR APPROACH)	**44.83**	**21.80**	**41.64**

Table 6.2: ROUGE-F1 (%) scores (with 95% confidence interval) of our *QExAbs* approach and various abstractive models on the Newsroom (Abstractive) test set. (*: taken from Grusky et al. (2018).)

Model	**R1**	**R2**	**RL**
SEQ2SEQ + ATTENTION * (Rush et al., 2015)	5.99	0.37	5.41
QEXABS (OUR APPROACH)	**17.13**	**2.71**	**15.41**

Table 6.3: ROUGE-F1 (%) scores (with 95% confidence interval) of our *QExAbs* approach and various abstractive models on the Newsroom (Mixed) test set. (*: taken from Grusky et al. (2018). †: taken from original paper.)

Model	**R1**	**R2**	**RL**
MODIFIED P-G * (Shi et al., 2019)	39.91	**28.38**	36.87
TLM * (Subramanian et al., 2019)	33.24	20.01	29.21
POINTER * (See et al., 2017)	26.04	13.24	22.45
TRANS-EXT+FILTER+ABS † (Egonmwan and Chali, 2019b)	35.74	16.52	30.17
QEXABS (OUR APPROACH)	**41.91**	23.66	**37.95**

6.5 Summary

In this chapter, we have described our second novel approach to generate query focused abstractive summaries. Here, we have shown that only the query-related sentences can be used for fine-tuning the pretrained language model (*BERTSUM*). Our experimental results show that our approach performs better on ROUGE for the CNN/Daily Mail and the *Abstractive* dataset of Newsroom. For the *Mixed* dataset of Newsroom, our approach performs better on ROUGE-1 and ROUGE-L, where on ROUGE-2, the model cannot perform better than the current state-of-the-art result.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we have generated queries for two publicly available datasets CNN/-Daily Mail and Newsroom according to the contexts of the documents and the summaries. We have designed an algorithm that can be used for generating queries for other summarization datasets. We have incorporated the query relevance on a recent state-of-the-art model (*BERTSUM*) with two different approaches *QAbs* and *QExAbs*. Our experimental results show that our approaches obtain new state-of-the-art ROUGE scores for CNN/Daily Mail and Abstractive dataset of Newsroom. For the Mixed dataset of Newsroom, both of our approaches show better results on ROUGE-1 and ROUGE-L. Results show that *QAbs* performs better on CNN/Daily Mail, and *QExAbs* performs better on Newsroom datasets shown in Table 7.1. Some sample summaries generated from our approaches are shown in Appendix A, where we can observe that our generated summaries hold almost same meaning as reference summaries.

7.2 Future Work

As we have achieved excellent results on our two novel approaches, in the future, we have a plan to do the following works to diversify our research:

- In our research, we considered only the common words between document and summary to generate the query. We will consider some similarity measures instead of considering the common words to generate the query.

Table 7.1: Comparison of the results between our implemented approaches.

Dataset	**Implemented Model**	**R1**	**R2**	**RL**
CNN/Daily Mail	BERTSUM	40.78	18.64	37.95
	QABS	**44.91**	**21.81**	**41.70**
	QEXABS	44.83	21.80	41.64
Newsroom (Abstractive)	BERTSUM	15.05	1.90	13.46
	QABS	15.05	2.26	13.50
	QEXABS	**17.13**	**2.71**	**15.41**
Newsroom (Mixed)	BERTSUM	32.09	15.21	28.88
	QABS	40.67	22.66	36.92
	QEXABS	**41.91**	**23.66**	**37.95**

- We will consider multiple documents for query focused abstractive summarization.
- We will try to use other pretrained language models by which we can find more appropriate models for abstractive summarization.
- To fine-tune the model, we will incorporate more datasets to improve the quality of summaries.

Bibliography

Rasim M Alguliyev, Ramiz M Aliguliyev, Nijat R Isazade, Asad Abdi, and Norisma Idris. 2019. Cosum: Text summarization based on clustering and optimization. *Expert Systems* 36(1):e12340.

Dave Anderson and George McNeill. 1992. Artificial neural networks technology. *Kaman Sciences Corporation* 258(6):1–83.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *CoRR* abs/1801.07704. http://arxiv.org/abs/1801.07704.

Facundo Bre, Juan M. Gimenez, and Víctor D. Fachinotti. 2018. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings* 158:1429 – 1441. https://doi.org/https://doi.org/10.1016/j.enbuild.2017.11.045.

Asli Çelikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *CoRR* abs/1803.10357. http://arxiv.org/abs/1803.10357.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, New York, New York, USA, IJCAI'16, page 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 484–494. https://doi.org/10.18653/v1/P16-1046.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 93–98. https://doi.org/10.18653/v1/N16-1012.

Morten H. Christiansen and Nick Chater. 2008. Language as shaped by the brain. *Behavioral and Brain Sciences* 31(5):489–509. https://doi.org/10.1017/S0140525X08004998.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116* .

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, pages 2978–2988. https://doi.org/10.18653/v1/P19-1285.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 4171–4186. https://doi.org/10.18653/v1/N19-1423.

L. Dong, S. Xu, and B. Xu. 2018. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pages 5884–5888.

Tamas E Doszkocs. 1986. Natural language processing in information retrieval. *Journal of the American Society for Information Science* 37(4):191–196.

Daniel M. Dunlavy, Dianne P. O'Leary, John M. Conroy, and Judith D. Schlesinger. 2007. Qcs: A system for querying, clustering and summarizing documents. *Information Processing & Management* 43(6):1588 – 1605. Text Summarization. https://doi.org/https://doi.org/10.1016/j.ipm.2007.01.003.

Madhurima Dutta, Ajit Kumar Das, Chirantana Mallick, Apurba Sarkar, and Asit K. Das. 2019. A graph based approach on extractive summarization. In Ajith Abraham, Paramartha Dutta, Jyotsna Kumar Mandal, Abhishek Bhattacharya, and Soumi Dutta, editors, *Emerging Technologies in Data Mining and Information Security*. Springer Singapore, Singapore, pages 179–187.

Elozino Egonmwan, Vittorio Castelli, and Md Arafat Sultan. 2019. Cross-task knowledge transfer for query-based text summarization. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Association for Computational Linguistics, Hong Kong, China, pages 72–77. https://doi.org/10.18653/v1/D19-5810.

Elozino Egonmwan and Yllias Chali. 2019a. Transformer and seq2seq model for paraphrase generation. In *Proceedings of the 3rd Workshop on Neural Generation and*

Translation. Association for Computational Linguistics, Hong Kong, pages 249–255. https://doi.org/10.18653/v1/D19-5627.

Elozino Egonmwan and Yllias Chali. 2019b. Transformer-based model for single documents neural summarization. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Association for Computational Linguistics, Hong Kong, pages 70–79. https://doi.org/10.18653/v1/D19-5607.

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

Changjian Fang, Dejun Mu, Zhenghong Deng, and Zhiang Wu. 2017. Word-sentence co-ranking for automatic extractive text summarization. *Expert Systems with Applications* 72:189 – 195. https://doi.org/https://doi.org/10.1016/j.eswa.2016.12.021.

Mohamed Abdel Fattah and Fuji Ren. 2009. Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Computer Speech & Language* 23(1):126 – 144. https://doi.org/https://doi.org/10.1016/j.csl.2008.04.002.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review* 47(1):1–66.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, ICML'17, page 1243–1252.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. *CoRR* abs/1808.10792. http://arxiv.org/abs/1808.10792.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 708–719. https://doi.org/10.18653/v1/N18-1065.

P. Gupta, V. S. Pendluri, and I. Vats. 2011. Summarizing text by ranking text units according to shallow linguistic features. In *13th International Conference on Advanced Communication Technology (ICACT2011)*. pages 1620–1625.

Johan Hasselqvist, Niklas Helmertz, and Mikael Kågebäck. 2017. Query-based abstractive summarization using neural networks. *CoRR* abs/1712.06100. http://arxiv.org/abs/1712.06100.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 199–208. https://doi.org/10.18653/v1/P17-1019.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1967–1972. https://doi.org/10.18653/v1/D15-1229.

Yacine Jernite, Samuel R Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557* .

Karen Sparck Jones and Brigitte Endres-Niggemeyer. 1995. Automatic summarizing. *Information Processing & Management* 31(5):625 – 630. Summarizing Text. https://doi.org/https://doi.org/10.1016/0306-4573(95)92254-F.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* .

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*. Association for Computational Linguistics, Vancouver, Canada, pages 67–72. https://www.aclweb.org/anthology/P17-4012.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, SIGIR '95, page 68–73. https://doi.org/10.1145/215206.215333.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR* abs/1901.07291. http://arxiv.org/abs/1901.07291.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* .

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2019. Flaubert: Unsupervised language model pre-training for french. *arXiv preprint arXiv:1912.05372* .

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv e-prints* page arXiv:1607.06450.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ILP for extractive summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1004–1013. https://www.aclweb.org/anthology/P13-1099.

Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th International Conference on World Wide Web*. Association for Computing Machinery, New York, NY, USA, WWW '09, page 71–80. https://doi.org/10.1145/1526709.1526720.

H. Van Lierde and Tommy W.S. Chow. 2019. Query-oriented text summarization based on hypergraph transversals. *Information Processing & Management* 56(4):1317 – 1338. https://doi.org/https://doi.org/10.1016/j.ipm.2019.03.003.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81. https://www.aclweb.org/anthology/W04-1013.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, pages 3730–3740. https://doi.org/10.18653/v1/D19-1387.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR* abs/1907.11692. http://arxiv.org/abs/1907.11692.

Yizhu Liu, Zhiyi Luo, and Kenny Zhu. 2018. Controlling length in abstractive summarization using a convolutional neural network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 4110–4119. https://doi.org/10.18653/v1/D18-1444.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893* .

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *CoRR* abs/1512.01712. http://arxiv.org/abs/1512.01712.

Chirantana Mallick, Ajit Kumar Das, Madhurima Dutta, Asit Kumar Das, and Apurba Sarkar. 2019. Graph-based text summarization using modified textrank. In Janmenjoy Nayak, Ajith Abraham, B. Murali Krishna, G. T. Chandra Sekhar, and Asit Kumar Das, editors, *Soft Computing in Data Analytics*. Springer Singapore, Singapore, pages 137–146.

Inderjeet Mani. 2001. *Automatic summarization*, volume 3. John Benjamins Publishing.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Xiangke Mao, Hui Yang, Shaobin Huang, Ye Liu, and Rongsheng Li. 2019. Extractive summarization using supervised and unsupervised learning. *Expert Systems with Applications* 133:173 – 181. https://doi.org/https://doi.org/10.1016/j.eswa.2019.05.011.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2019. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894* .

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'17, page 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Berlin, Germany, pages 280–290. https://doi.org/10.18653/v1/K16-1028.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 1797–1807. https://www.aclweb.org/anthology/D18-1206.

Preksha Nema, Mitesh M. Khapra, Anirban Laha, and Balaraman Ravindran. 2017. Diversity driven attention model for query-based abstractive summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers). Association for Computational Linguistics, Vancouver, Canada, pages 1063–1072. https://doi.org/10.18653/v1/P17-1098.

You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. 2011. Applying regression models to query-focused multi-document summarization. *Information Processing & Management* 47(2):227 – 237. https://doi.org/https://doi.org/10.1016/j.ipm.2010.03.005.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1949–1954. https://doi.org/10.18653/v1/D15-1226.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304. http://arxiv.org/abs/1705.04304.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 157–163. https://www.aclweb.org/anthology/E17-2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* .

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8):9.

Haggai Roitman, Guy Feigenblat, Doron Cohen, Odellia Boni, and David Konopnicki. 2020. Unsupervised dual-cascade learning with pseudo-feedback distillation for query-focused extractive summarization. In *Proceedings of The Web Conference 2020*. pages 2577–2584.

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, Lisbon, Portugal, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* .

Robert J Schalkoff. 1997. *Artificial neural networks*. McGraw-Hill Higher Education.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85 – 117. https://doi.org/https://doi.org/10.1016/j.neunet.2014.09.003.

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, Vancouver, Canada, pages 1073–1083. https://doi.org/10.18653/v1/P17-1099.

K. Shetty and J. S. Kallimani. 2017. Automatic extractive text summarization using k-means clustering. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*. pages 1–9.

Tian Shi, Ping Wang, and Chandan K. Reddy. 2019. LeafNATS: An open-source toolkit and live demo system for neural abstractive text summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 66–71. https://doi.org/10.18653/v1/N19-4012.

Shengli Song, Haitao Huang, and Tongxiao Ruan. 2019. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools Appl.* 78(1):857–875. https://doi.org/10.1007/s11042-018-5749-3.

Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. 2019. On Extractive and Abstractive Neural Document Summarization with Transformer Language Models. *arXiv e-prints* arXiv:1909.03186.

Krysta Svore, Lucy Vanderwende, and Christopher Burges. 2007. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, pages 448–457. https://www.aclweb.org/anthology/D07-1047.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, Las Vegas, pages 2818–2826.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1171–1181. https://doi.org/10.18653/v1/P17-1108.

Y. Tang, J. Xu, K. Matsumoto, and C. Ono. 2016. Sequence-to-sequence model with attention for time series classification. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. pages 503–510.

Wilson L. Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly* 30(4):415–433. https://doi.org/10.1177/107769905303000401.

Jorge V. Tohalino and Diego R. Amancio. 2018. Extractive multi-document summarization using multilayer networks. *Physica A: Statistical Mechanics and its Applications* 503:526 – 539. https://doi.org/https://doi.org/10.1016/j.physa.2018.03.013.

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

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence – video to text. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, USA, ICCV '15, page 4534–4542. https://doi.org/10.1109/ICCV.2015.515.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*. Association for Computing Machinery, New York, NY, USA, ICML '08, page 1096–1103. https://doi.org/10.1145/1390156.1390294.

Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, SIGIR '08, page 307–314. https://doi.org/10.1145/1390334.1390387.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzman, Armand Joulin, and Edouard Grave. 2019. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359* .

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*. pages 5754–5764.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

Appendix A

Sample System Generated Summaries

Here, we have shown the system generated summaries of our *QAbs* and *QExAbs* approaches.

A.1 CNN/Daily Mail Dataset

Reference Summary: agnese klavina , 30 , vanished after being driven away from a spanish club . older sister gunta made a video appeal for information on her whereabouts . brits westley capper and craig porter were summoned to court on monday .

***QAbs* generated summary:** agnese klavina , 30 , disappeared after a night out at aqwa mist in puerto banus . she was allegedly illegally detained by westley capper and co-accused craig porter . pair are currently being investigated on suspicion of agnese 's illegal detention .

***QExAbs* generated symmary:** agnese klavina , 30 , disappeared after a night out at aqwa mist in puerto banus . her boyfriend , michael millis , is a former owner of west london club westbourne studios . co-accused craig porter , 33 , also declined to answer questions during a closed court hearing in marbella . police have studied cctv footage from aqwa mist 's car park and a damning report by a criminal psychologist claiming her facial expressions show she did not leave the club voluntarily .

A.2 Newsroom Dataset

Reference Summary: a great beach selfie does n ' t make itself . a cool hairstyle helps .

***QAbs* generated summary:** a great selfie does n ' t make itself ; cool hair certainly helps . but it ' s not quite so simple . here ' s what you need to know about the beach with your look . use these four easy sunny - day style tips .

***QExAbs* generated symmary:** a great selfie does n ' t make itself ; cool hair certainly helps . but it ' s not quite so simple , says jean - sprin kaeling . (article plus video .) . (tips for getting ready . .