# Abstractive Summarization of Document using Dual Encoding Framework

Monika H. Rajput
PG Student
SSVPS's B.S. Deore College of Engineering,
Dhule, 424005, India

B. R. Mandre
Associate Professor
SSVPS's B.S. Deore College of Engineering,
Dhule, 424005, India

## ABSTRACT
Popularity of the web is increasing day by day and social media is becoming a huge source of information. It becomes difficult to analyze this enormous information quickly. Text summarization solves this problem, it minifies text such that repeated data are removed and important information is extracted and represented in the concise way which can help us to understand the information instantly. It is impossible to summarize all this information manually as it contains a huge number of unstructured information and reviews. Manual summarization is a tedious, monotonic and time consuming task. Therefore, method is needed for mining and summarizing information, reviews and produce representative summaries. To deal with this problem, an abstractive summarization of documents using an encoder decoder based approach is proposed. Abstractive Text Summarization gets the most essential content of a text corpus, compresses it to a shorter text, keeps its original meaning and maintains its semantic and grammatical correctness. For this, it uses deep learning architecture in natural language processing. It uses recurrent neural networks that connect the input and output data in encoder-decoder architecture with an added attention mechanism for better results. The proposed work is implemented with two datasets namely CNN/Dailymail and DUC 2004. It is worth mentioning that model achieves better performance than existing models, it improves result of ROUGE 1 metric to 41.75 for CNN/DailyMail and 35.12 for DUC2004.The experimental results show that the model produces a highly coherent, concise and grammatically correct summary.

## General Terms
Natural Language Processing, Encoder-decoder, Recurrent Neural Network, Summarization, Abstractive Summarization, Deep Learning

## Keywords
Summarization, Abstractive Summarization, Deep Learning, Natural Language Processing

## 1. INTRODUCTION
Summarization is a process of reducing text documents in order to create an accurate, concise and informative summary which conveys the main meaning of original text and retains important information of original document. Summarization is mainly classified into two approaches Extractive and Abstractive Summarization. Extractive summarization generates a summary by extracting relevant keywords, phrases and sentences from the original document. It has the advantage of grammatical and semantic correctness and disadvantage of redundancy and incoherence between sentences. Abstractive summarization generates the document summary by forming sentences on its own with the help of natural language generation techniques. Abstractive summarization uses readable human language to summarize key information from original documents. It creates an internal semantic representation of text and use this representation to create a more diverse and novel summary same as human written abstract. The proposed model focuses on an abstractive summarization.

Summarization is a sequence to sequence task for which a neural network based Encoder decoder model is used. In natural language processing, the neural network is widely used because of its promising performance. In text summarization, encoder reads variable length input sequence and produces a fixed dimensional feature vector from it. The decoder then uses this vector to produce output sequence [1].

In sequence to sequence network, when long dependencies are available in the model then it might be hard to summarize the whole input sentence into one single feature vector. Solution to this problem is the use of attention mechanism. At every decoding step, attention mechanism keeps the entire input sequence and uses it to produce output. It helps encoder and decoder to focus on useful information of input sequence. Decoder uses it to decide which parts of the source sequence to focus on, instead of forcing the encoder to compress all of the information into a single feature vector and passing it to the decoder [2]. This Recurrent Neural Network with attention mechanism approach performs better on many datasets but it cannot handle rare, unseen and Out of vocabulary (OOV) words effectively. Solution to this is Pointer mechanism (PM) which helps decoder to point back to OOV words and phrases and copy them directly into output [3]. Copying mechanism is used in the input text to derive the representations of OOV words from their corresponding context [4].

Repetition problems usually happen in case of long sequence generation tasks. During decoding, the coverage mechanism prevents decoder from attending the same part of the input sequence and eliminates frequently occurring phrases, words from longer summaries [5]. Decoded information at the decoder can also avoid repetition problems. Intra attention mechanism can also be effective for eliminating repetition [6]. But, all these consider little information about the relations between the input tokens in the encoder and the already generated words by decoder.

To solve all these problems, an abstractive summarization model is prepared using a dual encoding framework. This framework is an extension of the standard sequence to sequence framework. Framework consists of primary encoder, secondary encoder and decoder. Primary encoder reads an input sequence and produces a context vector for each word of the input sequence. Secondary encoder reweights remembered and forgotten parts in the input sequence. It calculates

importance weight for each word and recalculates the corresponding context vector. It uses input and previously produced output to generate a new context vector. This newly generated context vector is used by decoder to obtain more meaningful information and generate better output. To solve the repetition problem, a repetition avoidance mechanism (RAM) is used. RAM uses existing coverage mechanism with previously decoded content to reduce repetition problems in sequence to sequence tasks.

# 2. RELATED WORK

In 2015, Rush proposed work that uses a neural network based encoder decoder framework for an abstractive summarization. Model uses a convolutional neural network (CNN) as an encoder and a feed forward neural network as a decoder. It achieves good performance on DUC2003 and DUC2004 datasets [1].

In 2016, Chopra extends the work proposed by Rush, using Recurrent neural network for decoder in place of feed forward neural network [2]. Attention mechanism is added into this framework by Bahdanau to consider context cues in the hidden state of the encoder which facilitates decoding of the target sequence. Attention mechanism focuses on specific words at each step of the input sequence, determines the output, and emits the next word of the summary based on the previous ones. Attention mechanism solves the memorizing and representation problem of longer sequences of standard encoder decoder framework [8].

In 2016, Nallapati proposed a feature-rich hierarchical attentive encoder. It is based on the bidirectional GRU to represent the document. The hierarchical encoder has two RNNs. One RNN runs on the word-level and another runs on the sentence-level. The hierarchical attention re-weighs the word attention by the corresponding sentence-level attention. The weighted sum of the feature-rich input vectors is used to obtain document representation. The decoder is based on unidirectional GRU. The work uses the attention model on the hidden states of source and softmax layer on the target [3].

Input sequence consists of lots of rare words which are not available in target vocabulary. These rare, unseen and out of vocabulary (OOV) words prevent the model from learning new words during model training. In all previous work, a decoder uses a fixed target vocabulary to output the corresponding output at each time step. But it is unable to handle rare or OOV words. This problem can be solved by increasing the size of the target vocabulary, but this may increase the computational complexity in decoding as a softmax function needs to calculate over all possible words. This can be enhanced by applying a copy mechanism. At the time of decoding, the copy mechanism dynamically copies the words from the input sequence without enlarging the size of the vocabulary [3],[5],[7]. Zeng uses a copying mechanism to derive OOV words representation from their corresponding context vector in input text [4].

An encoder decoder model often generates unnatural summaries consisting of repeated phrases, especially in case of long text summarization. A coverage mechanism keeps track of words that have been already summarized. It avoids the repetition problem. It records past attentional weights in the decoder and prevents the decoder from attending to the same parts of the input text when decoding in future [3].

Paulus in 2018, proposed a deep reinforced model (DRM) for abstractive summarization. This model uses intra attention mechanism to eliminate repetition problems. In intra-attention mechanism, the decoder attends previously generated words. The main problem with this work is that they consider little about the relations between the input tokens in the encoder and the already generated words by decoder [6].

# 3. SYSTEM MODEL

Abstractive text summarization is a task of generating output sequence from input sequence. The input is a text sequence represented as $X=(x_1, x_2 \ldots x_m)$ where m is the number of words in the source text. Output is a shorter summary sequence such as $Y= (y_1, y_2 \ldots y_n)$ where n is number of words in summary text.

Proposed model consists of a primary encoder, secondary encoder and decoder furnished with an attention mechanism. Pointer mechanism and Repetition avoidance mechanism (RAM) are employed in Decoder. Pointer mechanism handles rare and OOV words of input sequence effectively. RAM avoids repetitive words problem of longer sequence generation task. The details of the primary encoder, secondary encoder and decoder are described in the below section.

## 3.1 System Architecture

Figure 1 represents a block diagram of a dual encoding model. It gives detail overview about the system. Three main blocks are Primary Encoder, Secondary Encoder and the decoder performs its intended task and makes working or execution easy.

1) The primary encoder block includes bidirectional Gated Recurrent Unit (GRU) as a recurrent unit. Primary encoder reads variable length input sequence, generates hidden state representation using bidirectional GRU. For each word position, Forward GRU sequentially computes hidden state representation and the backward GRU computes hidden representation in reversed sequence. Content representations for the whole input sequence is created by concatenating both hidden states representation and use this content to represent each word of input sequence. Primary encoder constructs a feature representation called a semantic vector for each word in the input sequence. Semantic vector contains hidden state representation and content representation.

2) The decoder is equipped with the attention mechanism that performs decoding operation in stages and generates partial fixed length output at each stage. Partial fixed length output contains decoded content representation.

3) Secondary encoder consists of a unidirectional GRU as a recurrent neural network. Hidden representation, content representation and decoded content representation is used by the secondary encoder to compute attention (importance) weights. Using these attention weights the corresponding context vector is recalculated and fed to the decoder to produce desired output. The secondary encoder does more fine encoding at each decoding stage. Decoder generates more accurate target output sequence using this new semantic context vector generated by secondary encoder.

Output of the primary and the secondary encoder are combined and fed to the decoder to produce a highly coherent summary which decreases repetition problem for longer sequence generation task.

### *3.1.1 Text Preprocessing*

Preprocessing is an important step and needs to be done before the actual analysis.

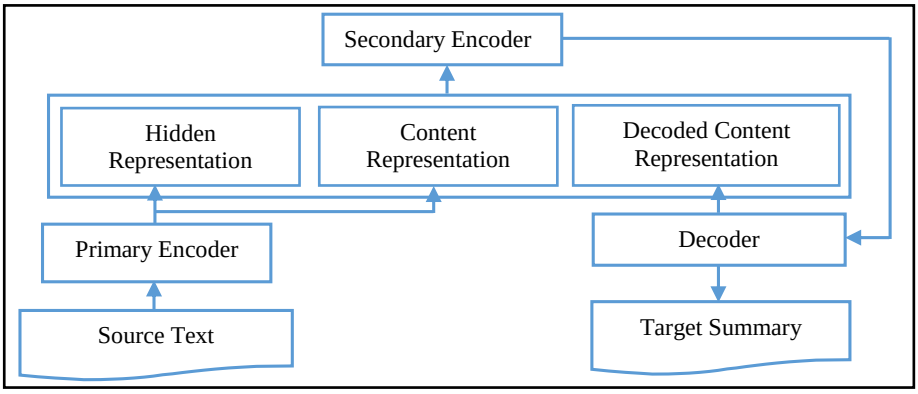


**Figure 1: Block Diagram of Dual Encoding Model**

Noisy, unstructured, irrelevant data can be major flaws in the dataset and increases the time of the training process. Preprocessed data improves the performance of the entire system. In preprocessing, unstructured data is converted into structured data format. The preprocessing consists of steps like dehtmlifying, tokenization, stopword removal, stemming, and Lemmatization and case normalization.

The data exist in HTML format so the first step in the preprocessing is to turn it into plain text and get rid of HTML tags. Tokenization can be done by removing punctuations and removing rare characters. Tokenization split sentences into words, usually white spaces, punctuations are used to delimit word boundaries. Then, these words can be simplified to reduce their number. At the end, tokens are obtained which is something more general than words.

In stopword removal, frequently occurring uninformative words are removed to speed up the processing. Stop words are words which occur very often in a language. For example, "a", "the" or "to" be considered as stop words for English language. Stemming trunks prefixes and suffixes from words and turns them into their stem forms that means to its root. Stemming reduces dimensions in word vector space and improves the algorithms. Lemmatization groups together words which are derivatives from the same word. For example, "eat, eats, eaten, eating, ate" can be reduced to a single token. Case Normalization converts all the characters to lowercase.

### 3.1.2 Primary Encoder

Proposed model employs a recurrent neural network as a primary encoder to handle variable length input sequences. A sequence of input word embedding ($x_1, x_2 \ldots x_m$) is provided to the primary encoder. Primary encoder performs encoding using Bidirectional Gated Recurrent Unit (GRU) based RNN. GRU has connections through sequences of nodes. At a different time scale, GRU captures dependencies in connection using Equation 1.

In Equation 1, $W_u$, $W_r$ and $W_h$ are parameter matrices. $x_t$ indicate the corresponding input embedding vector and $h_t$ is the hidden state vector at the time step t, and Θ is an element wise multiplication operator.

$$\begin{cases} \boldsymbol{h_t = (1 - u_t) \odot h_{t-1} + u_t \odot h'_t} \\ \boldsymbol{h'_t = tanh(W_u[x_t, r_t \odot h_{t-1}]} \\ \boldsymbol{r_t = \sigma(W_r[x_t, h_{t-1}])} \\ \boldsymbol{u_t = \sigma(W_u[x_t, h_{t-1}])} \end{cases} \quad \text{Equation 1}$$

Hidden state representation $(\vec{h}_1^p, \vec{h}_2^p, \ldots\ldots, \vec{h}_j^p, \ldots\ldots, \vec{h}_m^p)$ is sequentially computed using forward GRU for each word position using current word embedding and previous hidden state. The backward GRU for each word of input sentence computes hidden state $(\overleftarrow{h}_1^p, \overleftarrow{h}_2^p, \ldots\ldots, \overleftarrow{h}_j^p, \ldots\ldots, \overleftarrow{h}_m^p)$ representation in reversed order. Forward and backward hidden states are defined by Equation 2.

$$\begin{cases} \boldsymbol{\vec{h}_t^p = GRU^p(x_t, \vec{h}_{t-1}^p)} \\ \boldsymbol{\overleftarrow{h}_t^p = GRU^p(x_t, \overleftarrow{h}_{t-1}^p)} \end{cases} \quad \text{Equation 2}$$

A concatenated hidden state of forward and backward GRU as $h_t^p = [\vec{h}_t^p, \overleftarrow{h}_t^p]$ is used to represent each word in the input sequence. The content representation $C^P$ is calculated by using Equation 3.

$$\boldsymbol{C^P = \tanh\left(W_P \frac{1}{N} \sum_{t=1}^{N} h_t^p + b_p\right)} \quad \text{Equation 3}$$

In Equation 3, $W_p$ and $b_p$ are parameters, and N represents the length of the input sequence. For each iteration primary encoder using encoding mechanism reads each word in input sequence and generates the corresponding hidden state representation $h_j^p$ and content representation $C^P$. In the primary encoder, hidden representation is created only once in the model.

### 3.1.3 Secondary Encoder

Secondary encoder performs encoding using a unidirectional GRU based recurrent neural network. It reads the input sequence and computes importance weight $\alpha_t$ for every K decoding step according to decoded information of each stage. The importance weight $\alpha_t$ indicates how much attention should be paid to current input word $x_t$. Importance weight is computed by using Equation 4 based on feature representation of each word in input sequence, $C^P$ content representation of entire input sequence and $C^d$ content representation of entire output sequence generated by decoder. For each word in input sequence importance weight is computed based on information itself, its saliency and redundancy.

$$\boldsymbol{\alpha_t = \sigma\left(W_2\left(tanh\left(W_1[h_t^p, C^P, C^d] + b_1\right)\right) + h_t^{pT} W_s C^P + h_t^{pT} W_s C^d - C^{PT} W_r C^d + b_2\right)} \quad \text{Equation 4}$$

$W_1$, $W_2$, $W_s$, $W_r$, $b_1$, $b_2$ are learning parameters in Equation 4. Saliency between every word and the entire content of source text is modeled as $h_t^{pT} W_s C^P$ and $h_t^{pT} W_s C^d$ . $C^{PT} W_r C^d$ forms the Redundancy between the source text content and decoded content of current stage is formed by .

Importance weight is in the form of skip connection to bias the two information flows. If $\alpha_t$ value is approximates to 1, then it is similar to a standard GRU, which is only influenced

from the current word. If the current input word $x_t$ has a very small weight $\alpha_t$, then the hidden state $h_t^s$ is encoded by the secondary encoder which takes majority of information directly from the previous hidden $h_{t-1}^s$ and neglect the effect of the current word using Equation 5.

$$\boldsymbol{h_t^S = (1-\alpha)\odot h_{t-1}^S + \alpha_t \odot GRU^S(x_t, h_{t-1}^S)} \quad \text{Equation 5}$$

Secondary encoder does more fine encoding on the input sequence at every K decoding step and generates a new semantic context vector $h_m^S$. It facilitates the decoder to produce output more accurately. A combination of the primary and secondary encoder completes a dual encoding process.

### *3.1.4 Decoder*

The decoder and primary encoder forms a basic sequence to sequence model. To generate the output summary decoder uses GRU with an attention mechanism. Some advanced techniques are applied to model such as attention mechanism, copy mechanism, pointer generated network and coverage mechanism to achieve better performance of the model. A decoder with attention mechanism computes the context vector according to the hidden states $(h_1^p, h_2^p, \dots, h_j^p, \dots, h_m^p)$ of the primary encoder. The context vector $C_i$ is computed as a weighted sum of these hidden states using Equation 6.

$$\boldsymbol{C_i = \sum_{j=1}^{n} a_{ij} h_j^p} \quad \text{Equation 6}$$

$a_{ij}$ of Equation 6 is the weight which is computed for each hidden state by using Equation 7.

$$\begin{cases} \boldsymbol{a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n}\exp(e_{ik})}} \\ \boldsymbol{e_{ij} = v_a^T \tanh(W_a h_{i-1}^d + U_a h_j^p)} \\ \boldsymbol{h_i^d = GRU^d(y_i, h_{i-1}^d)} \end{cases} \quad \text{Equation 7}$$

$e_{ij}$ is a score that represents how well the inputs around position *j* match with the output at position *i*. The decoder generates hidden state $h_i^d$ based on its last hidden state $h_{i-1}^d$ and the $i^{th}$ target $y_i$ in the output sequence.

Dual encoding model does not decode the whole output sequence at one time but decodes the partial fixed-length sequence in stages using Equation 8.

$$\boldsymbol{C^d = \tanh\left(W_d \frac{1}{L}\sum_{i=1}^{L} h_i^p + b_d\right)} \quad \text{Equation 8}$$

In Equation 8, $W_d$ and $b_d$ are parameters and L is length of current decoded sequence. $C_d$ is the current content generated by the decoder. It adjusts attention weights of secondary encoder to each word in input sequence. A new final state $h_m^s$ is generated after every fixed length decoding and decoder is rewritten by using Equation 9.

$$\boldsymbol{h_i^d = \begin{cases} GRU^d(y_i, [h_{i-1}^d, h_m^s]) & \textbf{if L\%K == 0} \\ GRU^d(y_i, h_{i-1}^d) & \textbf{otherwise} \end{cases}} \quad \text{Equation 9}$$

Then, the current context vector $C_i$ produced from the primary encoder and the decoder hidden state are concatenated and feed to decoder through one linear layer to produce the vocabulary distribution using Equation 10.

$$\boldsymbol{P_v = P(y_i | y_1, \dots\dots, y_{i-1}; x) = softmax(W_v[h_i^d, C_i] + b_v)} \quad \text{Equation 10}$$

In this Equation 10, $W_v$, $b_v$ are learning parameters and $P_v$ is conditional probability distribution for the target word $y_i$ over all words in the vocabulary at time-step i.

#### *3.1.4.1 Pointer Mechanism*

Documents may contain some rare words or OOV words such as named-entities which prevents model from learning new words at training time. The proposed model handles such OOV words by simply pointing to their location in the source document. The PM is more robust in dealing with rare words as it uses the hidden state representation of rare words from the encoder to decide which word from the source document to point to. The hidden state depends on the entire context of the word so the model is still able to accurately point to unseen words which do not appear in the target vocabulary. Model uses a PM between the primary encoder and the decoder. Along with generating words from a fixed vocabulary, the model allows copying words via pointing.

A soft switch *Pp* helps the model to choose words from the fixed vocabulary or copy a word from the input sequence. A word is generated from the fixed vocabulary by sampling from $P_v$ or copy a word from the input sequence by sampling from the attention distribution $\alpha_i$. For time step i, a generation probability $P_p$ is calculated using Equation 11.

$$\boldsymbol{P_p = \sigma(W_c^T C_i + W_h^T h_i^d + W_y^T y_i + W_d^T C^d + b_g)} \quad \text{Equation 11}$$

In this Equation 11, $W_c$, $W_h$, $W_y$, $W_d$ and $b_g$ are learning parameters. $C_i$ is the context vector, $h_i^d$ is the decoder hidden state, $y_i$ is decoder input and $C^d$ is content representation of partial decoded sequence and σ is a sigmoid function.

Extended vocabulary is denoted as the union of fixed vocabulary and all the words appearing in source document. The probability distribution over the extended vocabulary is calculated using Equation 12. If w is an OOV word then $P_v(W)=0$ and $\sum_{j:w_j=w} a_{ij}$ is also zero.

$$\boldsymbol{P_w = P_p P_v(W) + (1-P_p)\sum_{j:w_j=w} a_{ij}} \quad \text{Equation 12}$$

During training $\mathcal{L}_i = -\log P(W_i)$ computes the loss for time step i which is given by the negative log likelihood of the target word $W_i$. The overall loss for the whole sequence is computed as the average of all losses at each time step, which is given by Equation 13. In this, T denotes length of target sequence.

$$\boldsymbol{\mathcal{L} = \frac{1}{T}\sum_{i=0}^{T} \mathcal{L}_i} \quad \text{Equation 13}$$

#### *3.1.4.2 Repetition Avoidance Mechanism*

Repetition is a common problem in sequence generation tasks such as summarization, mostly in generating multi sentence text summary. The coverage mechanism aims to deal with the repetition problems from the encoder by discouraging the decoder from attending to the same part of the input sequence according to the past attentional weights. Combination of the coverage mechanism and content produced in the earlier time steps by the decoder, avoids the repetition from both the encoder and the decoder. The proposed model uses an enhanced mechanism to solve this problem. The secondary encoder first generates an encoding of the feature vector at every K step. This allows the decoder to remember the content produced in the earlier time steps to avoid the repetition then it uses coverage mechanism [5]. The coverage vector $C^v$ is defined as the sum of attention distributions over all previous decoder time steps using Equation 14.

$$\boldsymbol{C_i^v = \sum_{i'=0}^{i-1} a_i'} \quad \text{Equation 14}$$

The coverage vector is also used as extra input to the attention mechanism in Equation 7. So the formula is updated as

$$e_{ij} = v_a^T \tanh\left(W_a h_{i-1}^d + U_a h_j^p + W_{ch} C_i^v\right) \quad \text{Equation 15}$$

An additional coverage loss is defined to penalize repeatedly attending to the same locations to avoid repetition. Combining coverage with Equation 13, the primary loss function is rewritten as

$$\mathcal{L} = \frac{1}{T}\sum_{i=0}^{T}\left(\mathcal{L}_i + \lambda \sum_j min\left(a_{ij}, C_{ij}^v\right)\right) \quad \text{Equation 16}$$

In this Equation 16, λ is a hyper parameter. i is the decoding time step and j denotes the position in the input sequence.

## 3.2 Model Training Algorithm

**Input:** Preprocessed dataset which contains text to summarize (X) and reference summaries (Y).

**Output:** Summary Sentences

1. Given Training set <X,Y>
2. **for** episode=0, M **do**
3. Sample (x,y) from source text X and gold summary Y
4. Compute the hidden state of primary encoder $h_t^p$ for each word in x Eq. (1) and Eq. (2)
5. Compute the content representation $C^P$ for *x* using Eq. (3)
6. **for** decoding time-step *i* = 0, *len(Y)* **do**
7. Compute the hidden state of decoder $h_i^d$ using Eq. (7)
8. **if** *i*%*K* == 0 **then**
9. **if** *i* == 0 **then**
10. Set the content representation of partial generated Sequence $C^d$ to zero
11. **else**
12. Compute $C^d$ using Eq. (8)
13. **end if**
14. Compute the importance weight $\alpha_t$ using Eq 4)
15. Compute the hidden state of secondary encoder $h_t^s$ using Eq. (5)
16. Compute the hidden state of decoder $h_i^d$ based on $h_{i-1}^d$ and $h_m^s$ in Eq. (9)
17. **end if**
18. Compute the vocabulary distribution $P_w$ using Eq. (12)
19. Update network parameters based on the overall loss $\mathcal{L}$ in Eq. (16)
20. **end for**
21. **end for**

Figure 2 represents step by step working of proposed model.

# 4. EXPERIMENTAL RESULTS

Proposed model uses a lot of training data so it requires GPU to handle data efficiently. The proposed model is implemented using Github's atom IDE, Tensorflow and various packages provided by python. The model converges on the machine with a 2.3 GHz Intel Xenon processor, 16 GB memory and NVIDIA Tesla T4 GPU card coupled with 16 GB memory.

## 4.1 Dataset

Experiment is performed on two challenging datasets CNN/DailyMail and DUC2004. CNN/DailyMail dataset is used to train and test the proposed model mostly multi sentence summarization. DUC2004 is used as a testing dataset to evaluate performance of proposed model.

**CNN/DilyMail**

CNN/DailyMail dataset contains a collection of articles mostly news, interviews that have been published on the two popular websites CNN.com and dailymail.com. This dataset originally has been gathered in the work of Hermann [9] for the question answering task, has become a standard source for training and evaluating text summarizer models and later modified for abstractive summarization task and has been used in many studies. In this dataset, there are on an average 28 sentences per document in the training set, and an average of 3 ~ 4 sentences in the reference summaries. The dataset contains 2,86,817 examples in the training set, 13,368 examples in the validation set, and 11,487 examples in testing set.

**DUC 2004**

Document Understanding Conference DUC-2004 is one of the most used evaluation datasets for summarization task. It contains 500 articles issued by The New York Times and Associated Press paired. It also contains corresponding reference summaries written by four humans.

## 4.2 Experimental Settings

Proposed model has 3 main modules. The preprocessing module, training, and evaluation module. Each has a wide set of hyper parameters and is highly customizable. All the hyper parameters have a default value. The training module defines and customizes proposed models. These modules also define the training procedure and adjust the optimization parameter values. To train and test the model batch size is set to 32. The dimension of hidden states of both encoders and decoder are set to 512 and 768 respectively. The size of the vocabulary is limited to 50,000 by selecting the most frequent tokens in the training set. OOV words are represented as a token. The dimension of word embedding is set to 128 which are learned during training. Learning rate is set to 0.15. The decoding length is set to 20 for the CNN/DailyMail dataset and 10 for DUC2004 dataset. At the testing time, the same length settings have used and decoded the output summaries using beam search with beam size 5.

## 4.3 Methods to Compare

In this work, to compare the performance of proposed model some advanced methods of summarization are used.

**CNN/DailyMail Dataset**

1) ***words-lvt2k-temp-att*** – It is an abstractive encoder decoder based model with the temporal attention mechanism. Temporal attention mechanism keeps track of past attentional weights of the decoder and restricts the repetitive parts in the later sequence [3].

2) ***pointer-generator*** – It is a standard sequence to sequence an attentional model based on a hybrid pointer generator network. This hybrid pointer generator deals with rare or OOV words problem [5].

3) ***pointer-generator+coverage*** –"pointer-generator" model is extended by addition of a coverage mechanism to avoid the repetition and denoted by "pg+cg" [5].

4) ***RL+ML*** – It is a new training approach for abstractive summarization using a neural network with intra attention and Pointer generator mechanism [6].

5) ***seq2seq+atten*** – It is a standard sequence-to-sequence encoder-decoder model employed with attention mechanism for abstractive text summarization [8].

6) ***SummaRuNNer-abs*** – It is a RNN based model for abstractive summarization. It is converted from an extractive model by using a novel training mechanism [10].

All above approaches including the proposed model are the standard supervised sequence prediction model using maximum-likelihood training except hybrid training method.

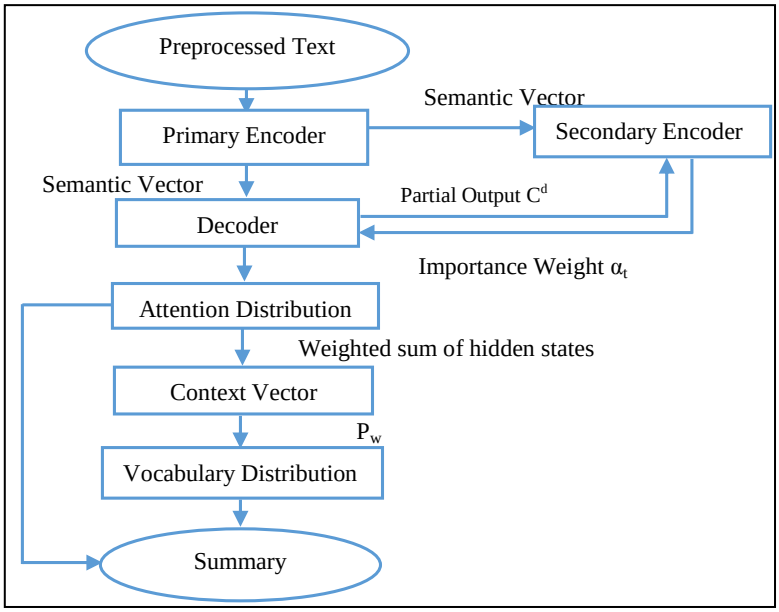


**Figure 2: Work Flow diagram of dual encoding model**

**DUC2004 Dataset**

1) ***ABS*** **-** It uses a local attention-based mechanism to generate each word of the summary [1].

2) ***ABS+*** **-** It combines a conventional ABS and an additional log linear extractive summarization model with hand crafted features [1].

3) ***RAS-Elman*** **-** In this, summarization is performed using an attentive encoder and RNN based decoder [2].

4) ***words-lvt5k-lsent*** – It is an attentional encoder decoder model and uses large vocabulary tricks for summarization [3].

5) ***TOPIARY*** **-** It uses a linguistically motivated compression method and an unsupervised topic detection algorithm for summarization [11].

6) ***SEASS*** **-** It is a selective encoding model with a selective gate network. It controls the information flow from encoder to decoder and constructs a second level sentence representation [12].

## 4.4 Evaluation Metrics

In this work, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric is used to automatically evaluate the performance of the proposed model and compare it with other reported performances in the literature. ROUGE is the standard metric used to evaluate summary qualities. It calculates a score for a generated summary based on how well the summary overlaps with a set of golden truth summary references. A loss is defined based on this ROUGE score. For the good summary, the ROUGE score should be high and the loss should be less. ROUGE-1 matches unigram, ROUGE-2 matches bigrams and ROUGE-L matches longest common subsequences between reference summaries and system generated summaries. Pyrouge package provided by python is used to calculate ROUGE score.

## 4.5 Results on CNN/DailyMail Dataset

Table 1 shows experimental results of various model on CNN/DailyMail testing dataset. Here, decoding length is set to 20 to compare results across various models. The proposed model achieves state-of-art performance compared to others. Use of Pointer mechanism, RAM and dual encoding helps model to improve results and produce a better summary.

Figure 3 shows performance comparison of various methods across proposed model in graphical format.

**Table 1 Performance comparison of various model on CNN/DailyMail dataset using F1 score**

| Method | ROUGE 1 | ROUGE 2 | ROUGE L |
|---|---|---|---|
| **Seq2seq+atten** | 31.34 | 11.79 | 28.10 |
| **Words-lvt2k-temp-att** | 35.46 | 13.3 | 32.65 |
| **SummaRuNNer-abs** | 37.5 | 14.5 | 33.4 |
| **Pointer-generator** | 36.44 | 15.66 | 33.42 |
| **RL+ML** | 39.87 | 15.82 | 36.90 |
| **Pg+cg** | 39.53 | 17.28 | 36.38 |
| **DEATS** | 40.85 | 18.08 | 37.13 |
| **Proposed System** | **41.75** | **19.44** | **38.81** |

In all above compared methods, only the "pg+cg" approach uses a coverage mechanism to prevent the repetition. Proposed model uses enhanced repetitions avoid mechanism which combines the coverage mechanism with the previously generated output of decoder to improve the quality of the generated summary. All the compared methods conduct just one encoding process on the input sequence to generate the complete target summary in one step, while proposed method uses a dual encoding with multistep decoding operation. Specifically, the secondary encoding in the proposed model performs fine and selective encoding based on the input and the previous output that helps the decoder produce a better summary.

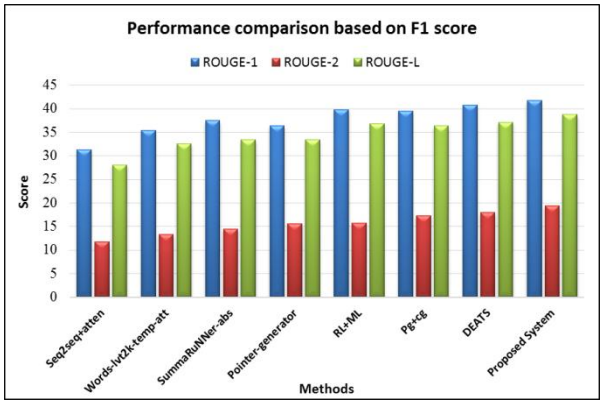


**Figure 3: Performance comparison of various models on the CNN/DailyMail testing set using ROUGE F1 score**

To evaluate the influence of different decoding lengths on the performance, length is set to K = {10, 15, 20, 25, 30, 40, 50, 100}. Model calculates ROUGE-1, ROUGE-2 and ROUGE-L Recall, Precision and F1 score for different decoding lengths**.** Table 2 shows experimental results of ROUGE precision score and Table 3 shows results of ROUGE recall score for different decoding lengths on CNN/DailyMail dataset.

**Table 2 Performance comparison of proposed system for different decoding lengths on the CNN/DailyMail testing set using rouge precision score**

| Metric → / Length ↓ | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| **10** | 40.74 | 19.06 | 37.89 |
| **15** | 40.74 | 19.06 | 37.89 |
| **20** | 40.73 | 19.05 | 37.89 |
| **25** | 40.72 | 19.05 | 37.87 |
| **30** | 40.69 | 19.02 | 37.84 |
| **40** | 40.49 | 18.90 | 37.66 |
| **50** | 39.87 | 18.56 | 37.07 |
| **100** | 32.63 | 14.89 | 30.40 |

**Table 3 Performance comparison of proposed system for different decoding lengths on the CNN/DailyMail testing set using rouge recall score**

| Metric → / Length ↓ | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| **10** | 45.85 | 21.28 | 42.60 |
| **15** | 45.86 | 21.28 | 42.60 |
| **20** | 45.86 | 21.28 | 42.61 |
| **25** | 45.89 | 21.29 | 42.63 |
| **30** | 45.93 | 21.30 | 42.67 |
| **40** | 46.16 | 21.39 | 42.89 |
| **50** | 46.95 | 21.72 | 43.63 |
| **100** | 55.53 | 25.35 | 51.76 |

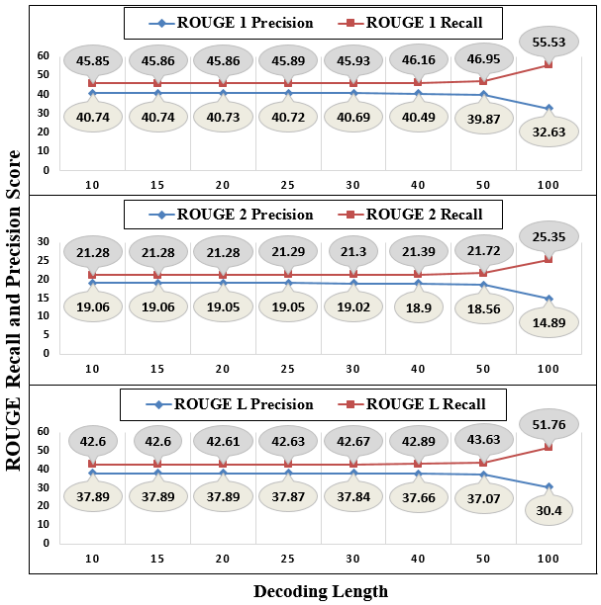


**Figure 4: ROUGE Recall vs Precision score of proposed models on the CNN/DailyMail testing set**

According to results in Table 1 and Table 5, proposed model achieves more improvement in performance as compared to DEATS model for both CNN/DailyMail and DUC2004 dataset. Repetition phenomenon affects the performance of the generated summary which is well handle by the proposed model. Secondary encoder of proposed model conducts a finer encoding and helps model to consider richer and more accurate information. Figure 4 shows comparison of ROUGE recall vs ROUGE precision of proposed model for different decoding lengths on CNN/DailyMail testing set. Figure shows higher precision and lower recall results for different decoding lengths.

**Table 4 Performance comparison of proposed system for different decoding lengths on the CNN/DailyMail testing set using rouge F1 score**

| Length | ROUGE-1 | | ROUGE-2 | | ROUGE-L | |
|---|---|---|---|---|---|---|
| | Proposed System | DEATS | Proposed System | DEATS | Proposed System | DEATS |
| **10** | 41.74 | 38.26 | 19.44 | 16.44 | 38.80 | 34.90 |
| **15** | 41.74 | 40.35 | 19.44 | 17.57 | 38.81 | 36.72 |
| **20** | 41.75 | 40.85 | 19.44 | 18.08 | 38.81 | 37.13 |
| **25** | 41.76 | 40.62 | 19.44 | 17.37 | 38.82 | 37.37 |
| **30** | 41.72 | 40.71 | 19.44 | 18.18 | 38.83 | 37.23 |
| **40** | 41.82 | 40.76 | 19.43 | 17.99 | 38.86 | 37.20 |
| **50** | 41.88 | 40.75 | 19.42 | 17.52 | 38.93 | 37.16 |
| **100** | 39.98 | 39.85 | 18.23 | 17.45 | 37.26 | 36.58 |

Table 4 shows the experimental result of ROUGE F1 score for different decoding lengths on CNN/DailyMail dataset. Table compares result of proposed system vs result of DEATS model [13]. Results shows model achieves more improvement in performance compared to DEATS model.

**Table 5 Performance comparison of various model on DUC 2004 dataset using Recall score**

| Method | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| **TOPIARY** | 25.12 | 6.46 | 20.12 |
| **ABS** | 26.55 | 7.06 | 22.05 |
| **ABS+** | 28.18 | 8.49 | 23.81 |
| **RAS-Elman** | 28.97 | 8.26 | 24.06 |
| **words-lvt5k-lsent** | 28.61 | 9.42 | 25.24 |
| **SEASS** | 29.21 | 9.56 | 25.51 |
| **DEATS** | 29.91 | 9.61 | 25.95 |
| **Proposed Method** | **35.12** | **15.49** | **28.82** |

Table 5 shows the experimental result of various models on DUC2004 testing set. Decoding length is set to 10 for DUC2004 dataset. Proposed model achieves more improvement in performance as compared to DEATS model. Figure 5 shows performance comparison of various models across proposed model in graphical format.

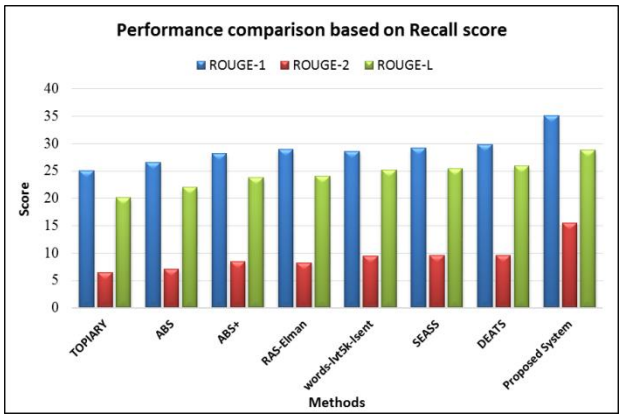


**Figure 5: Performance comparison of various models on the DUC2004 testing set using ROUGE F1 score**

Table 6 shows experimental results of ROUGE recall score for different decoding lengths on DUC2004 dataset.

**Table 6 Performance comparison of proposed system for different decoding lengths on the DUC 2004 testing set using rouge recall score**

| Metric → / Length ↓ | ROUGE-1 | ROUGE-2 | ROUGE-L |
|---|---|---|---|
| **10** | 35.12 | 15.49 | 28.82 |
| **15** | 35.63 | 15.65 | 29.16 |
| **20** | 37.05 | 16.09 | 30.1 |
| **25** | 39.37 | 16.82 | 31.51 |
| **30** | 41.77 | 17.54 | 32.84 |
| **40** | 45.43 | 18.70 | 34.92 |
| **50** | 47.76 | 19.43 | 36.33 |
| **100** | 52.16 | 21.22 | 39.45 |

Figure 6 shows comparison of ROUGE recall vs precision of proposed model for different decoding lengths on DUC 2004 testing set. Figures shows higher precision and lower recall when decoding length is set to a smaller value. When decoding length is set to larger value it shows opposite results. When decoding length is set to 20-30, model achieves good tradeoff between recall and precision.

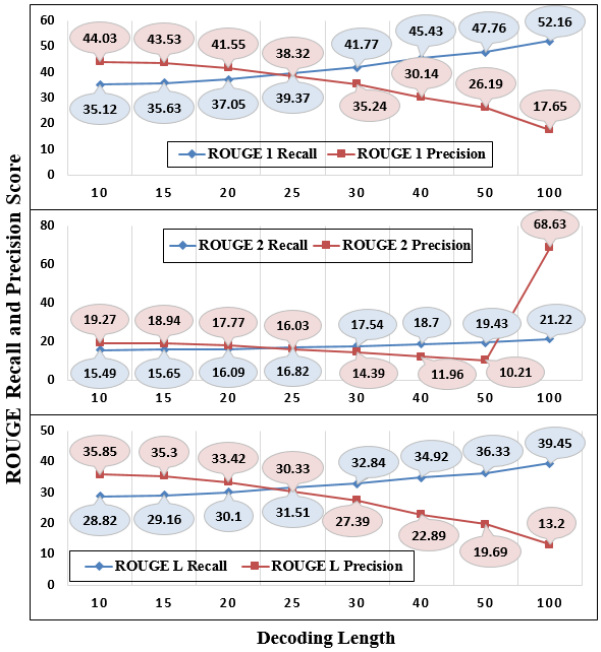


**Figure 6: ROUGE Recall vs Precision score of proposed models on the DUC 2004 testing set**

## 5. CONCLUSION

Abstractive Summarization using a dual encoding framework is an extension of the standard sequence to sequence encoder decoder model with attention mechanism, Pointer mechanism, Repetition avoidance mechanism and multistage decoding. Pointer Mechanism of decoder handles rare and out of vocabulary words from input text. It helps the model to learn new words during training and increases readability of the model without increasing vocabulary size. Repetition avoidance mechanism of decoder remembers the content produced in the earlier time-step, which helps model to avoid repetition problems. Dual encoding decodes the whole output sequence by stages and produces partial fixed length sequence at each stage which helps model to tackle problems of existing methods. Dual encoding with all basic approaches produces highly coherent and more accurate summary. It is worth mentioning that model achieves better performance than existing models, it achieves improved ROUGE 1 score 41.75 for CNN/DailyMail and 35.12 for DUC2004. The proposed dual encoding model performs state of art results on CNN/DailyMail and DUC2004 dataset.

## 6. REFERENCES

[1] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP), Lisbon, Portugal, Sep. 2015, pp. 379–389.

[2] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in Proc. Conf. North Amer. Assoc. Comput. Linguist. Human Lang. Technol., San Diego CA, USA, Jun. 2016, pp. 93–98.

[3] R. Nallapati, B. Zhou, C. N. dos Santos, Ç Gülçehre, and

B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in Proc. 20th SIGNLL Conf. Comput. Nat. Lang. Learn. (CoNLL), Berlin, Germany, Aug. 2016, pp. 280–290.

[4] W. Zeng, W. Luo, S. Fidler, and R. Urtasun, "Efficient summarization with read-again and copy mechanism," CoRR, vol. abs/1611.03382, 2016.

[5] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2017, pp. 1073–1083.

[6] Romain Paulus, Caiming Xiong, and Richard Socher, "A deep reinforced model for abstractive summarization". In Proceedings of the 6th International Conference on Learning Representations, Vancouver, Canada, 2018.

[7] P. Li, W. Lam, L. Bing, and Z. Wang, "Deep recurrent generative decoder for abstractive text summarization," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2091–2100.

[8] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End to end attention-based large vocabulary speech recognition," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), Shanghai, China, Mar. 2016, pp. 4945–4949.

[9] K. M. Hermann et al., "Teaching machines to read and comprehend," in Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process.Syst., Montreal, QC, Canada, Dec. 2015, pp. 1693–1701.

[10] R. Nallapati, F. Zhai, and B. Zhou, "SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents," in Proc. 31st AAAI Conf. Artif. Intell, San Francisco, CA, USA, Feb. 2017, pp. 3075–3081.

[11] D. Zajic, B. Dorr, and R. Schwartz, "Bbn/umd at DUC-2004: Topiary," in Proc. Doc. Understanding Conf. NLT/NAACL, 2004, pp. 112–119.

[12] Q. Zhou, N. Yang, F. Wei, and M. Zhou, "Selective encoding for abstractive sentence summarization," in *Proc. Meeting Assoc. Comput. Linguist*, 2017, pp. 1095–1104.

[13] Kaichun Yao, Libo Zhang, "Dual Encoding for Abstractive Text Summarization", IEEE TRANSACTIONS ON CYBERNETICS, China, 2018 pp. 2168-2180.