Automatic Text Summarization

Ayasha Agrawal 184101008
Megha Jain 184101024
Monika Khandelwal 184101026
Shavi Gupta 184101033

Abstract

Our model is a neural network based sequence model for automatic text summarisation, it achieves performance that is comparable to state of the art model. Our model predicts summary that has features such as information content, salience and novelty.

1 Introduction

Text summarisation is a process of condensing a document to its shorter version that content important information related to the document.The graph based approach for text summarisation suffers from inconsistencies, lack of balance, results in lengthy summaries.Our model which uses neural network learn various features to produce summary.

2 Methods

2.1 Using GRU-RNN Model

We can model extractive summarization as Sequence Classification problem that is visit every sentence sequentially in original document and take binary decision whether to include the sentence in the summary.
The basic building block for model is gated recurrent unit (GRU) RNN. GRU-RNN has 2 gates:
u: update gate which decides how much of current information to use in updating the cell state.
r: reset gate which decides how much of the past information to forget.
It can be described by following equations:

$$u_j = \sigma(W_{ux}x_j + W_{uh}h_{j-1} + b_u)$$
$$r_j = \sigma(W_{rx}x_j + W_{rh}h_{j-1} + b_r)$$
$$h_j^{'} = tanh(W_{hx}x_j + W_{hh}(r_j \odot h_{j-1}) + b_h)$$
$$h_j = (1 - u_j) \odot h_j^{'} + u_j \odot h_{j-1}$$

where W's and b's are weights and bias of the GRU-RNN
h_j: vector with real values that represents hidden-state vector at time step j
$h_j^{'}$: vector with real values that represents candidate hidden-state vector at time j
x_j: input vector at time step j
$\odot$: Hadamard product

Phase 1: Create document representation
Our model comprises of 2 layers of Bi-directional GRU-RNN.
Layer 1: It comprises of a pair of forward and backward GRU-RNNs that run at word level in forward and backward direction respectively and sequentially compute the hidden state representation at each word position taking into consideration the current word embeddings and previous hidden state.
Layer 2: It comprises of a pair of forward and backward GRU-RNNs that run at sentence level in forward and backward direction respectively. It accepts average-pooled or max-pooled concatenated hidden states of Layer 1 and sequentially compute the hidden state representation for each sentence of the document.
The entire document can be represented as:

$$d = tanh(W_d \frac{1}{N_d} \sum_{j=1}^{N^d} [h_j^f, h_j^b] + b)$$

where
N_d denotes number of sentences in the document , $'[]'$ denotes vector concatenation and h_j^b and h_j^f denotes hidden state representation for jth sentence with respect to backward and forward sentence-level GRU-RNN.

Phase 2: Classification
In the 2nd phase, we will revisit each sentence sequentially and decide whether sentence should belong to summary by calculating probability of being part of summary for each sentence based on certain parameters. We will use linear/ Bi-linear transformation models to:

1. Estimate Information Content of sentence: It will take as input the hidden representation of sentence h_j and calculate the information content of sentence.
   Content can be computed as $W_c h_j$

2. Estimate Salience: It calculates importance or prominence of the sentence with respect to the document by taking as input the hidden representation of sentence h_j and document representation d.
   Salience can be computed as $h_j^T W_s d$

3. Estimate Redundancy: It calculates the redundancy of the sentence with respect to the current state of summary s_j that is the candidate summary computed until jth sentence.
   Redundancy can be computed as $h_j^T W_r tanh(s_j)$

4. Estimate Absolute position importance: It calculates the importance of absolute position of sentence with respect to the document using absolute position vector p_j^a as input. Thus we will need to find absolute position vector embedding for our model.
   It can be computed as $W_{ap} p_j^a$

5. Estimate Relative position importance: It calculates the importance of relative position of sentence with respect to the document. Relative position is calculated by dividing the entire document into segments and considering the segment ID to which sentence belongs as its relative position. We will also need to find relative position vector embedding for our model.
   It is can be computed as $W_{rp} p_j^r$

Thus the probability that the binary label y_j associated with the sentence at jth position given by hidden state representation h_j has value 1, that is the sentence is included in the summary can be given as:

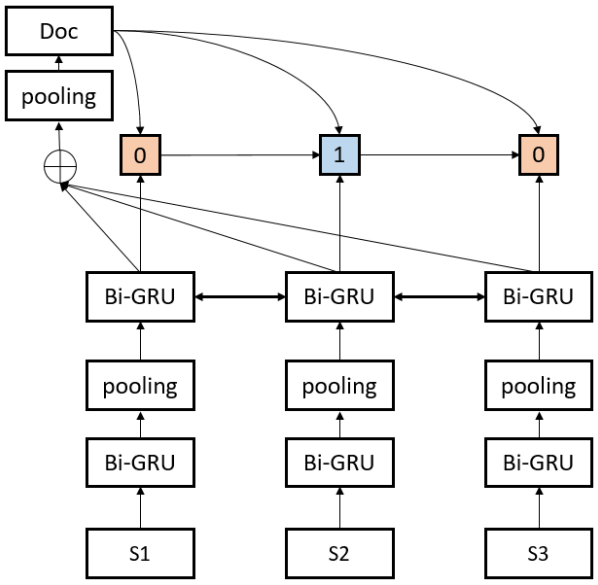

Figure 1: Two-layer GRU-RNN Model

$$P(y_j = 1/h_j, s_j, d) = \sigma(W_c h_j + h_j^T W_s d - h_j^T W_r tanh(s_j) + W_{ap} p_j^a + W_{rp} p_j^r + b)$$

where
d denotes document representation
s_j denotes current state summary at jth sentence position and is given by:

$$s_j = \sum_{i}^{j-1} h_i P(y_i = 1/h_i, s_i, d)$$

p^a denotes absolute positional embedding p^r denotes relative positional embedding

2.2 Using Attention Model

In RNN_RNN model the representation of sentences in a vector form is done by taking average pooling of the vector representation of all the words of that sentence. This makes the importance of every word to be same. Whereas in attention_RNN model the contribution of the more relevant words are more than the less relevant words means it pays attention to the more relevant word.

Attention in text summarization means that the words in the output summary are being related to some specific words of input document.We have used hierarchical attention model because we need to connsider the more relevannt sentences inn a document and also the more relevant words

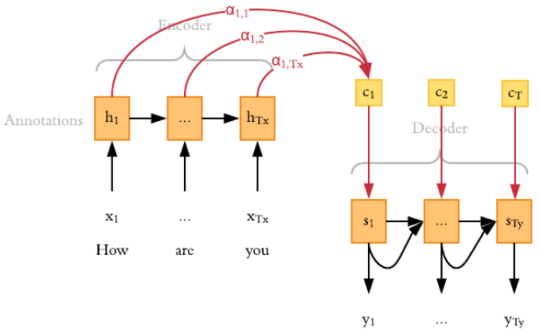

Figure 2: Internal working of Attenntion Model: the attention model creates a separate context vector for each output word inspite of converting the entire input sequence into a single context vector.

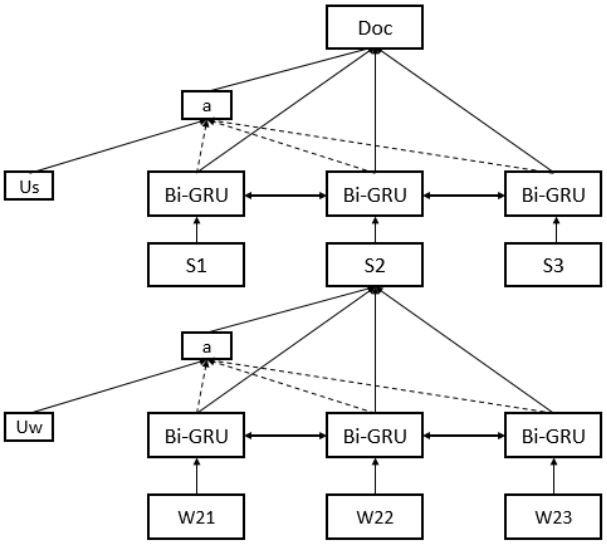

Figure 3: Attention RNN :

of the sentence. This layer is added after generating vectors for words that is word_RNN(x) and also after generating the vector representation for sentences that is sent_RNN(x).

Inputs:

query : tensor containing the query features from the decoder.

context : tensor containing features of the encoded input sequence.

Outputs:

query : tensor containing the attended query features from the decoder.

attn : tensor containing attention weights.

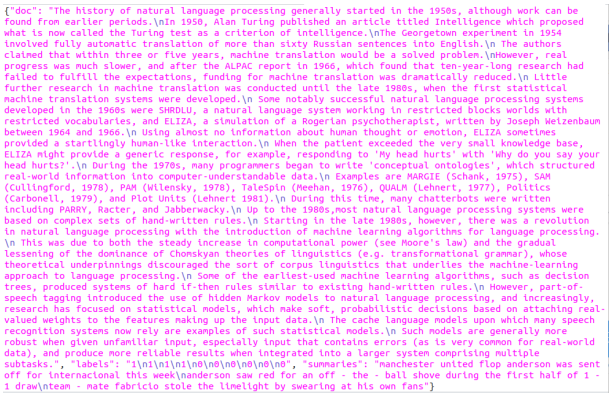

Figure 4: Input document

2.3 Training of Model

During training the model, we estimate the parameters that maximize the log-likelihood of the observed labels in the training data set.Thus, our loss function l(W,b) becomes:

$$l(W,b) = -\sum_{d=1}^{N}\sum_{j=1}^{N_d}(y_j^d log P(y_j^d = 1/h_j^d, s_j^d, d_d)$$

$$(1 - y_j^d) log(1 - P(y_j^d = 1/h_j^d, s_j^d, d_d))$$

where

N denotes total number of documents

N_d denotes number of sentences in the document

y denotes vector of binary labels for each sentence.

Parameters of GRU-RNN model:

1. Weights and Bias used in both phases
2. Absolute and Relative positional embedding

Hyper-Parameters of GRU-RNN model:

1. Kernel size used for average or max-pooling
2. Number of segments in which we divide the document.

During testing, the model gives probability of membership for summary for each sentence.The top k lines with highest probability are selected as extractive summary for the document.

The document[Figure 4] and corresponding generated summary are shown [Figure 5] [Figure 6]:

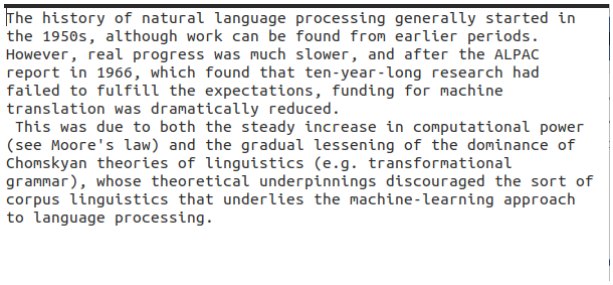

Figure 5: Summary generated through RNN Model

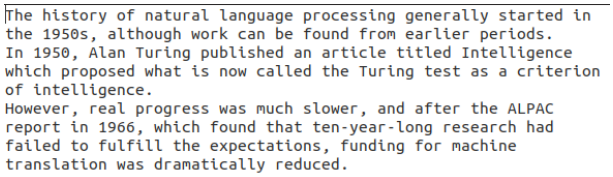

Figure 6: Summary generated through Attension

3 Experimental Design

Dataset:
We are using the corpus CNN/DailyMail for the task of extractive text summarization.In this corpus there are 196,557 training documents, the validation set is of the size 12,147 and the test document is of size 10,396. If we consider CNN subset also,then our we will have 286,722 training documents, 13,362 validation documents and 11,480 test documents. In training set, each document contains :

- Text: Sequence of sentences for which summary has to be extracted(approx 805 words)
- Labels: Binary variable corresponding to each sentence the document with value 1 if that sentence is part of summary else value 0.
- Summary for the text (approx 3-4 lines)

4 Result

The evaluation of model is done by calculating Rouge_1 metrics, calculated with respect to reference summary.The output yielded contains:

1. Precision : how much overlapping the generated summary with the reference summary .
   We can calculate it by ratio of number of overlapping word to number of words in generated summary.(If the reference and generated summary is same than precision value is 1)

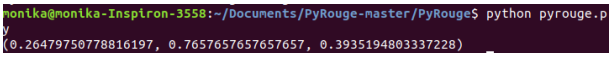

Figure 7: Rouge score output of one random document

2. Recall :how much overlapping the reference summary with the generated summary .
   We can calculate it by ratio of number of overlapping word to number of words in reference(ideal) summary.

3. F Score : It is harmonic average of precision and recall.

The performance of model on DailyMail Corpus is shown in table below:

Model	F-Score
RNN-RNN	0.258
Attension Model	0.26

The rouge score of above document summary is shown in [Figure 4].

5 Conclusion

We have used a simple NN model to implement text summarisation for single document and present them in precise format.In our approach we have first used RNN based model for summarisation and then moved to Attention based neural network model by which the rouge score of summary has improved.

6 References

- A. Nenkova, and K. McKeown, "Automatic summarization,". Foundations and Trends in Information Retrieval, 5(2-3):103233, 2011.
- R. Nallapati, F. Zhai and B. Zhou, "SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents,". In AAAI, 2017.
- A Statistical Approach for Document Summarization