Text Summarization Project: Bengali abstractive summarization using deep learning

Mitodru Niyogi
Interdisciplinary Center for Scientific Computing, Heidelberg University
Heidelberg, Germany
niyogi@cl.uni-heidelberg.de

Abstract

Text summarization using deep learning has become quite a well studied research problem for high resource languages such as English and other European languages. However, very few works have been done for poorly available resource languages such as Indian subcontinent languages, African languages across the Internet. The resource scarce languages have limited scope in natural language processing (NLP) so far with lack of proper parallel corpus, parser, chunkers, tokenizer, POS taggers, etc. In this paper, we propose an abstractive text summarization sequence to sequence deep learning model for Bengali. We performed end-to-end word-level cross entropy training of the sequence to sequence model on Bengali news articles scrapped from a major daily news paper in Bangladesh. We also trained Word2vec on the dataset to generate Bengali word embeddings. The model used is based on encoder-decoder architecture. The encoder has multilayered Bidirectional Long Short-Term Memory (LSTM) layers to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. The decoder consists of a uni-directional LSTM-RNN with the same hidden-state size, and an attention mechanism over the source-hidden states and a soft-max layer over target vocabulary to generate abstractive summary sentences. We have evaluted the summary generation using ROUGE-1, ROUGE-2, ROUGE-L scores to evaluate the deep learning model. We also presented and analyzed both the good quality and bad quality summary generated by our model. We also used beam size of 10 for beam search decoder to generate summaries.

Keywords: Abstractive Summarization, Seq2Seq model, Bahdenau attention mechanism, Bengali NLP, deep learning

1. Introduction

Rcently, deep learning based sequence-to-sequence models have been quite successful in many computer vision, and natural language processing problems such as machine translation (Bahdanau et al. (2014)), speech recognition (Bahdanau et al. (2015)) and video captioning (Venugopalan et al. (2015)). In this project, we focus on the task of abstractive text summarization for resource scare Indian languages such as Bengali news documents using deep learning models. Generally, text summarization can also be considered as mapping of large input sequence of words in a source document to a relative shorter length target sequence of words called summary, keeping the main content very briefly. In the framework of sequence-to-sequence models, a very relevant model to our task is the attentional RNN encoder-decoder model proposed in Bahdanau et al. (2014), which has produced state-of-the-art performance in machine translation (MT). In the modern era of big data, retrieving useful information from a large number of textual documents is a challenging task, due to the unprecedented growth in the availability of blogs, news articles, and reports are explosive. Automatic text summarization provides an effective solution for summarizing these documents. The increasing amount of non-English data available through the Internet and processed by several modern age IR/NLP (natural language processing) tasks has magnified the importance of summarizing multilingual documents in order to quickly gain information exchange in non English world especially Indian languages manifold. In countries such as India where there are 22 official languages with 600 million Internet users, multiple languages are used officially and regularly by a large amount of computer-educated citizens, the above task is particularly important, and can be a game changer for many of the digital initiatives that governments across the world are actively promoting.

2. Background

2.1. Word Embeddings

Given a sentence contains N words $S = \{x_1, x_2, ..., x_N\}$. Every word x_i is projected into a high dimensional real-valued vector e_i Each word x_i is transformed into word embedding e_i by using the matrix-vector product

$$e_i = W^{wrd} v^i \qquad (1)$$

where v^i is a vector of size $|V|$ which has the value 1 at index e_i and 0 in all other positions, W^{wrd} is the embedding matrix, V is the vocabulary size. Then the sentence is feed into the next layer as a real-valued vectors $embs = \{e_1, e_2, ..., e_T\}$.

2.2. Badaneu Attention mechanism

Badaneu attention mechanism is an additive attention as it performs a linear combination of encoder and decoder states. Attention layer consists od alignment layer, attention weights, and context vector. The alignment score maps how well the inputs around position “j” and the output at position “i” match. The score is based on the previous decoder's hidden state, s_{i-1} just before predicting the target word and the hidden state, h_j of the input sentence. Then a softmax activation is applied on the alignment scores to produce the attention weights. The decoder decides which part of the source sentence it needs to pay attention to, instead of having encoder encode all the information of the source sentence into a fixed-length vector. The context vector is used to compute the final output of the decoder. The context vector c_i is the weighted sum of attention weights and the encoder hidden states $(h_1, h_2, ..., h_{tx})$, which maps to the input sentence.

2.3. Seq2Seq model

Seq2Seq model was firstly used in machine translation [Bahdanau et al. (2014)] to translate one sequence to another sequence through an encoder-decoder neural architecture . After that it became popular in abstractive summary generation where it has been treated as sequence translation from an original text to a summary [Rush et al. (2015); See et al. (2017); Paulus et al. (2017)]. In our project, we have used bidirectional Long Short-term Memory Network as an encoder architecture and RNN decoder with Badaneu Attention mechanism. In a text summarization problem, given a article text $X = \{x_1, x_2, ..., x_L\}$ represented into a sequence of word embeddings (Word2vec, Glove, etc.), the text encoder receives the token of words in x and encodes them into a series of context vectors $H = (h_1, h_2, ..., h_L)$ though a bidirectional Long Short-term Memory Network (BiLSTM). The BiLSTM captures the contextual information from past and future words into the vector representation h_t of a particular word vector x_t , as follows:

$$h_t = \overrightarrow{h_t} + \overleftarrow{h_t} \qquad (2)$$

$$h_t = \overrightarrow{LSTM}(x_t, \overrightarrow{h_{t-1}}) \qquad (3)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(x_t, \overleftarrow{h_{t+1}}) \qquad (4)$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden outputs of the forward LSTM and the backward passes of the BiLSTM respectively.
And then, the decoder sequentially generates a summary $Y = \{y_1, y_2, ..., y_M\}$ using context vectors as input, as defined below:

$$p(Y|X) = \sum_{t=1}^{M} p(y_t|c_t, y_1, ..., y_{t-1}) \qquad (5)$$

At t-th time step, the decoder RNN generates one word conditioned on the attentive context vector c_t and the decoder hidden state s_t. The generation probability of the t-th word can be calculated as:

$$p(y_t|X) = softmax(W_g u_t) \qquad (6)$$

$$u_t = \tanh(W_{st} s_t + W_{ct} c_t) \qquad (7)$$

$$s_t = \overrightarrow{LSTM}(c_t, y_{t-1}, s_{t-1}) \qquad (8)$$

where y_{t-1}, s_{t-1} are the last generated words and the hidden state of decoder LSTM at (t-1)-th time step. W_g, W_st and W_{ct} are parameter matrices. The attention mechanism computes an attentive context vector c_t at t-th time step given the context vector, which allows the decoder to give attention not just on the final hidden encoder state but on every hidden states of the encoder as weighted mean of encoder hidden states. c_t is computed as follows:

$$c_t = \sum_{i=1}^{N} \alpha_{ti} h_i \tag{9}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{N} \exp(e_{tj})} \tag{10}$$

$$e_{tj} = \tanh(s_{t-1}^T W_t h_j) \tag{11}$$

where c_t is a weighted sum of context vector h_i in H, and the weight α_{ti} for the each h_i is sequentially computed by Equation 10 and 11; the W_t is a trainable parameter matrix.

2.4. Beam Search

Beam search is a graph search algorithm that uses breadth-first search to build its search tree by generating sequences from left to right. At each level of the tree, it generates all successors of the states at the current level and sorts them in increasing order of heuristic cost. However, it only retains a predetermined number, B top scoring , of best states at each level (called the beam width). Only top B states are expanded next. The larger is the beam width, the fewer states are pruned. If the beam width is of infinite size then no states are pruned and beam search is identical to breadth-first search. The beam width bounds the memory required to perform the search. Since a goal state could potentially be pruned, beam search sacrifices completeness (the guarantee that an algorithm will terminate with a solution, if one exists). Beam search is not optimal (that is, there is no guarantee that it will find the best solution). This idea has also been applied to neural machine translation (NMT) to model the bi-directional dependency of source and target texts.
In general, beam search returns the first solution found. In case of machine translation/text summarization tasks, the beam search after reaching the configured maximum search depth (i.e. translation length), the algorithm will evaluate the solutions found during search at various depths and return the best one (the one with the highest probability).
The beam width can either be fixed or variable. One approach that uses a variable beam width starts with the width at a minimum. If no solution is found, the beam is widened and the procedure is repeated.

3. Dataset

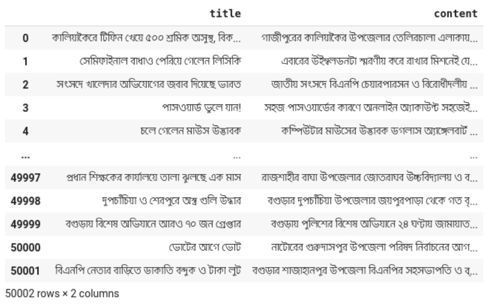
The dataset[1] is a part of a Kaggle dataset in json format. The dataset contains Bengali news articles scrapped from a major leading Bengali daily newspaper, Prothomalo [2] in Bangladesh. A full training dataset has the following attributes:

- author: author of the news article
- category: category name in English
- category_bn: category name in Bengali
- published_date: the publication date of the article
- modification_date: modification date of the article
- tag: broader categorization of the article
- comment_count: count of the total number of comments on the article
- title: the title of a news article
- url: url source of the article
- content: the content of the news articles

The training dataset has 400k+ bangla news samples across 25+ categories annotated articles related to politics, sports, economy, etc.

[1]https://www.kaggle.com/furcifer/bangla-newspaper-dataset

[2]https://www.prothomalo.com/



	title	content
0	কালিয়াকৈরে টিফিন খেয়ে ৫০০ শ্রমিক অসুস্থ, বিক...	গাজীপুরের কালিয়াকৈর উপজেলার তেলিরচালা এলাকায়...
1	সেমিফাইনাল বাধাও পেরিয়ে গেলেন লিসিকি	এবারের উইম্বলডনটা স্মরণীয় করে রাখার মিশনেই যে...
2	সংসদে খালেদার অভিযোগের জবাব দিয়েছে ভারত	জাতীয় সংসদে বিএনপি চেয়ারপারসন ও বিরোধীদলীয় ...
3	পাসওয়ার্ড ভুলে যান!	সহজ পাসওয়ার্ডের কারণে অনলাইন অ্যাকাউন্ট সহজেই...
4	চলে গেলেন মাউস উদ্ভাবক	কম্পিউটার মাউসের উদ্ভাবক ডগলাস অ্যাঙ্গেলবার্ট ...
...	...	...
49997	প্রধান শিক্ষকের কার্যালয়ে তালা ঝুলছে এক মাস	রাজশাহীর বাঘা উপজেলার জোতরাঘব উচ্চবিদ্যালয় ও ব...
49998	দুপচাঁচিয়া ও শেরপুরে অস্ত্র গুলি উদ্ধার	বগুড়ার দুপচাঁচিয়া উপজেলার জয়পুরপাড়া থেকে গত বৃ...
49999	বগুড়ায় বিশেষ অভিযানে আরও ৭০ জন গ্রেপ্তার	বগুড়ায় পুলিশের বিশেষ অভিযানে ২৪ ঘণ্টায় জামায়াত...
50000	ভোটের আগে ভোট	নাটোরের গুরুদাসপুর উপজেলা পরিষদ নির্বাচনের আগ...
50001	বিএনপি নেতার বাড়িতে ডাকাতি বন্দুক ও টাকা লুট	বগুড়ার শাজাহানপুর উপজেলা বিএনপির সহসভাপতি ও ব...

50002 rows × 2 columns

Figure 1: Snapshot of dataset

3.1. Data Preprocessing

We have used a subset of the large dataset for our experiments and the title field, and the content have been kept in the subset dataset. This dataset has 50000 data points in which we have used 80-20% split for training and validation split. The title has been considered as a summary of the document whereas the content field has been considered as article text. The following steps were taken:

- Removal of English characters, punctuations, digits, urls, latin symbols and unwarranted characters
- Removal of Bengali punctuations, period (।)
- Removal of Bengali stopwords were optional (অবশ্য, অনেক, অনেকে, অনেকেই, অন্তত, আপনি)
- Tokenization and building of Bengali vocabulary dictionary
- Gensim Word2vec skipgram model with the configuration of size=300, window size=10, min_count=1 were trained on the dataset to generate Bengali word vectors of dimension 300
- Finally we have sorted the summaries and article to the extent that max(len(text)) = 400, max(len(summary))= 30, min(len(summary)) = 5, max(unk count of text)= 20, max(unk count of summary) =10. This results in the dataset consisting of 24213 article-summary pairs for training and validation purposes.

3.2. Prepreocessing for Seq2Seq model:

We have included some special tokens in vocabulary such as <UNK>, <PAD>, <EOS>, <GO>. Vocabulary is limited for some reasons. UNK token for missing words in the vocabulary. PAD token adds a batch size of each sentence has the same length. EOS token marks the end of the sequence for input to the encoder. GO token to mark the start the process of output sequence in the decoder. Before train data, we chose GO and EOS in data which contains words id used sequence translation. In this, the sequence to sequence mode x is input sequence of encoder and y are generated output or response output sequence

3.3. Dataset statistics

The size of the vocabulary is 312294. The number of <UNK> tokens in the text is 312 which is basically less than 1% of total number of tokens. The total number of words in the summaries is 242723 and the total number of tokens in artcle text is 12897153.

Table 1: Length statistcs of summary and article

Length	**Text**	**Summary**
Mean	1676.888144	31.60
Std.	1515.820531	13.30
25%	767	22.00
50%	1306	32.00
75%	2092.750	41.00
Max	46084.	660.00

Table 2: Word count

Field	**#tokens**
Text	12897153
Summary	242723

In the next section, we introduce the training and decoding process for summarization.

4. Model

4.1. Training strategy

The aim of the word-level training for language models is to optimize predictions of the next token (Ranzato et al. (2015)). For example, in the abstractive text summarization, a seq2seq model generates a summary y with the probability $P_\theta(y|x)$ given a source article x as input, where θ represents model parameters (e.g., weights W and bias b). In a neural language model (Bengio et al.2003), $P(y_t \mid y_{<t}, x)$, known as likelihood, i.e, the conditional probability of the next token y_t given all previous tokens denoted by $y_{<t} = (y_1, y_2, ..., y_{t-1})$ and source x. Intuitively, the text generation process can be described as follows:

- The decoding process starts with a special token '<GO>' (start of sequence), the model generates a token y_t at a time t with the probability $P(y_t \mid y_{<t}, x) = P_{vocab,t}(y_t)$.
- This token can be obtained by greedy search, i.e., $y_t = \arg\max_{y_t} P_{vocab,t}$ or by a sampling method.
- The generated token is fed into the next decoding step. The generation continues unless whenthe model outputs 'EOS' (end of sequence) token or when the user defined maximum threshold length is reached.

We used end-to-end cross-entropy training to learn the model parameters, i.e., θ. 1) Cross-Entropy Training (XENT) (Ranzato et al. (2015)): To learn model parameters θ, XENT maximizes the log-likelihood of observed sequences (ground-truth) $\hat{y_t} = (\hat{y_1}, \hat{y_2}, ..., \hat{y_T})$, i.e.,

$$\log P_\theta(\hat{y}|x) = \sum_{t=1}^{T} \log P_\theta(\hat{y}|\hat{y}_{<t}, x) \qquad (12)$$

5. Experiments

In this section, we discuss about the training details, hyperparameters tunning, decoding and evaluation of generated summaries.

5.1. Training

Training: For all the models we discuss below, we used 300-dimension word-embeddings pre-trained by the word2vec algorithm (Mikolov et al. (2013)) on the Bengali news dataset

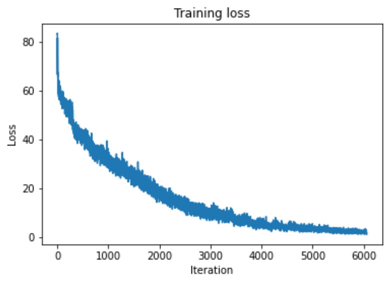


Figure 2: Training loss (20 epochs, 3 layers Encoder)

that we used in our experiments,and we allowed the embeddings to be updated during training. The encoder consists of a bidirectional LSTM-RNN, each with a hidden state dimension of 400. We varied the number of BiLSTM encoder layers between 2 and 3 in our experiments. The decoder consists of a unidirectional LSTM-RNN with the same hidden-state size, an attention mechanism over the source-hidden states and a soft-max layer over target vocabulary to generate words. The decoder output is limited to the user-defined maximum length of the summary. We kept the source and target vocabularies separate for computational efficiency, since the target vocabulary is much smaller. We used Adam optimizer for training, with an initial learning rate of 0.001. We used a batch-size of 32, 64 and randomly shuffled the training data at every epoch. We also used dropout probability and varied it between 0.5 and 0.7. We also applied gradient clipping to reduce the gradient explosion of RNNs networks We used early stopping based on the validation set and used the best model on the validation set to report all performance numbers. We used beam size of 10 for beam search decoder to generate summaries.

5.1.1. Hyperparameters tunning

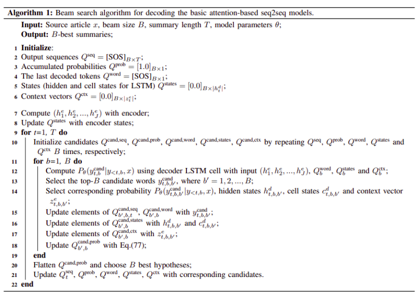
We varied the hyperparameters such as number of epochs, number of RNN layers, batch size, and dropout probability in our experiments. From the table 5 we see that the model with 3 layers have given higher Rouge scores than the model with 2 layers. We also notice that with the increase of the number of epochs and a dropout probability of 0.5 and batch size of 64 has resulted in higher rouge scores than the rest of the other paramters tuning.

5.2. Decoding and evaluation

At decode-time, we used beam search of size 10 to generate the summary, and limit the size of summary to a maximum of 30 words, as the maximum length of the summary is 62 in the dataset. We report Precision, Recall, F1-scores from the full-length version of Rouge-1, Rouge-2, and Rouge-L using the official evaluation script. Similar to (Nallapati et al.,2016) and (Chopra et al., 2016), we use the full length F1 variant of Rouge to evaluate our system. There is another preferred metric known as limited length recall but it limits in choosing the length which varies from corpus to corpus varies from corpus to corpus, making it difficult for comparisons among the researchers. On the other hand, Full-length recall, does not impose a length restriction but unfairly favors longer summaries. Full-length F1 solves this problem since it can penalize longer summaries, while not imposing a specific length restriction

5.3. Generating summaries

An approach is to approximate the arg max with a strictly greedy or deterministic decoder. A compromise between exact and greedy decoding is to use a beam-search decoder (Algorithm 1) for inference which considers the full vocabulary V while limiting itself to k potential hypotheses at each position of the summary. It iteratively consider the set of the k best sentences up to time t as candidates to generate sentences of size t + 1, and keep only the resulting best k of them. The beam-search algorithm is shown here, modified for the feed-forward model:



```
Algorithm 1: Beam search algorithm for decoding the basic attention-based seq2seq models.
  Input: Source article $x$, beam size $B$, summary length $T$, model parameters $\theta$;
  Output: $B$-best summaries;
Initialize:
Output sequences $Q^{\text{seq}} = [\text{SOS}]_{B\times T}$;
Accumulated probabilities $Q^{\text{prob}} = [1.0]_{B\times 1}$;
The last decoded tokens $Q^{\text{word}} = [\text{SOS}]_{B\times 1}$;
States (hidden and cell states for LSTM) $Q^{\text{states}} = [0.0]_{B\times |h_t^d|}$;
Context vectors $Q^{\text{ctx}} = [0.0]_{B\times |z_t^e|}$;

Compute $(h_1^e, h_2^e, ..., h_J^e)$ with encoder;
Update $Q^{\text{states}}$ with encoder states;
for $t$=1, $T$ do
    Initialize candidates $Q^{\text{cand,seq}}$, $Q^{\text{cand,prob}}$, $Q^{\text{cand,word}}$, $Q^{\text{cand,states}}$, $Q^{\text{cand,ctx}}$ by repeating $Q^{\text{seq}}$, $Q^{\text{prob}}$, $Q^{\text{word}}$, $Q^{\text{states}}$ and
      $Q^{\text{ctx}}$ $B$ times, respectively;
    for $b$=1, $B$ do
        Compute $P_\theta(y_{t,b}^{\text{cand}}|y_{<t,b}, x)$ using decoder LSTM cell with input $(h_1^e, h_2^e, ..., h_J^e)$, $Q_b^{\text{word}}$, $Q_b^{\text{states}}$ and $Q_b^{\text{ctx}}$;
        Select the top-$B$ candidate words $y_{t,b,b'}^{\text{cand}}$, where $b' = 1, 2, ..., B$;
        Select corresponding probability $P_\theta(y_{t,b,b'}^{\text{cand}}|y_{<t,b}, x)$, hidden states $h_{t,b,b'}^d$, cell states $c_{t,b,b'}^d$ and context vector
          $z_{t,b,b'}^e$;
        Update elements of $Q_{b',b,t}^{\text{cand,seq}}$, $Q_{b',b}^{\text{cand,word}}$ with $y_{t,b,b'}^{\text{cand}}$;
        Update elements of $Q_{b',b}^{\text{cand,states}}$ with $h_{t,b,b'}^d$ and $c_{t,b,b'}^d$;
        Update elements of $Q_{b',b}^{\text{cand,ctx}}$ with $z_{t,b,b'}^e$;
        Update $Q_{b',b}^{\text{cand,prob}}$ with Eq.(77);
    end
    Flatten $Q^{\text{cand,prob}}$ and choose $B$ best hypotheses;
    Update $Q_t^{\text{seq}}$, $Q^{\text{prob}}$, $Q^{\text{word}}$, $Q^{\text{states}}$, $Q^{\text{ctx}}$ with corresponding candidates.
end
```

Figure 3: Beam Search Algorithm

Computational costs: We trained all our models on a single Tesla K80/Tesla P100-PCIE-16GB GPU available on Google Colab. In Colab, random GPU gets available so different models were trained on different available GPUs. Therefore comparing different models on time requirements are not ideal and thus we skip the comparison. However, the training took about 10 hours to train for 20 epochs. Generating summaries at test time is reasonably fast with a throughput of about 10 summaries per second on a single GPU, using a batch size of 1.

6. Results

We report the F1 score, precision, recall from the full length version of Rouge-1, Rouge-2, Rouge-L on the validation set. We have listed both good and poor quality summaries generated by the seq2seq model in the following tables 3, and 4 respectively. The best summaries have average Rouge score above 60%. We see that in the best cases, the model has been able to infer the context of the article to replicate the summary as given in the reference summary. Interestingly, the poorly generated summaries with poor ROUGE scores of less than 25% have ot been able to summarize the article but reflected a statement or as an after cause of an event discussed in the article. For eg. the real summary: "Suchitra Sen is under ventilation." discusses about the critical condition, and hospital admission of "Suchitra Sen" whereas the generated summary says about post hospital admission fact as "Suchitra Sen recovers and returns in household activities." In another example, we can see that the generated summary (Bathing of tree improves health) doesn't make any sense to the ground truth which talks about the bathing of a child in winter days. Upon visual inspection of the system output, we noticed that on this dataset, both these models produced summaries that contain repetitive phrases or even repetitive sentences at times. Since the summaries in this dataset involve multiple sentences, it is likely that the decoder "forgets" what part of the document was used in producing earlier highlights.

Table 3: Generated good summaries

Reference Summary	Generated Summary
বিএনপির স্থায়ী কমিটির বৈঠক আজ কাল জোটের (The committee meeting of BNP will take place in a day)	বিএনপির স্থায়ী কমিটির বৈঠক কাল (The committe meeting of BM will take place tomorrow)
আগুনে ১৫টি দোকান পুড়ে গেছে (15 shops gutted in fire)	অগ্নিকাণ্ডে ৯ দোকান পুড়ে ছাই (9 shops turned into ashes in fire (due to fire))
সাগর রুনির মামলার আপাতত অগ্রগতি নেই স্বরাষ্ট্র প্রতিমন্ত্রী (Home Minister says that there is no current progress now in Sagar Runi case.)	সাগর রুনির মামলার অগ্রগতি নেই (There is no progress in Sagar Runi case)
হরতালের প্রতিবাদে গণজাগরণ মঞ্চের মিছিল (Rally by Ganajagaran Manch in protest of strike)	চট্টগ্রামে গণজাগরণ মঞ্চের বিক্ষোভ মিছিল (Protest rally by Ganajagaran Manch in Chattagam)
শ্রীনগর বিএনপির ৭৭ নেতা কর্মীর জামিন (77 Leaders and party workers of BNP released in Srinagar.)	বিএনপির দুই নেতা কর্মীর বিরুদ্ধে শুরু (tarts against 2 leaders and party workers of BNP)

Table 4: Generated poor summaries

Reference Summary	Generated Summary
শীতে কি শিশু গোসল করবে না (Will not a child take bath in winter?)	গাছের গোসল করে সুস্থ বুঝতে বাধ্য (Bathing of tree improves health)
সুচিত্রা সেনকে ভেন্টিলেশনে রাখা হয়েছে (Suchitra Sen is under ventilation.)	সুচিত্রা সেনের বাড়ির কাজে ফেরা (Suchitra Sen recovers and returns in household activities.)
আদমদীঘির মিতইল প্রাথমিক বিদ্যালয়ে ককটেল বিস্ফোরণ (Cocktail explosion at Mitil Primary School, Adamdighi.)	বগুড়ায় হলো ককটেল হামলায় একজন আটক (One person held in cocktail attack in Bagurah.)
চকরিয়ায় মৎস্য ঘেরে লুটপাট আহত ৩ (Arson in fish market at Chakriya, 3 wounded)	চকরিয়ায় ১৮ দলের কার্যালয়ে আগুন (Fire at 18 party Offices in Chakriya)

7. Conclusion and Future Scope

Being one of the most successful applications of seq2seq models, neural abstractive text summarization has become a prominent research topic that has gained a lot of attention from both industry and academia. In this paper, we have tried to address the problem of abstractive text summarization for poor resource South Asian language, Bengali by using sequence-to-sequence encoder-decoder deep learning architecture. We also evaluated the summary generation using popular ROUGE metric. We also analyzed the good and bad abstractive summaries generated. To our best knowledge, this is first of its kind work where deep learning model has been properly used for Bengali abstractive summarization on Prothomalo Bengali news dataset. In future, we would like to address the repetition handling problem by using pointer based copying mechanism or using state-of-the-art BERT models for abstractive text summarization.

8. References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473.

[Bahdanau et al.2015] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2015. End-to-end attentionbased large vocabulary speech recognition. CoRR, abs/1508.04395.

[Rush et al.2015] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 379–389.

[Ranzato et al.2015] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," arXiv preprint arXiv:1511.06732, 2015.

[Bengio et al.2003] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," Journal of machine learning research, vol. 3, no. Feb, pp. 1137–1155, 2003.

[Chopra et al.2016] S. Chopra, M. Auli, and A. M. Rush, "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, 2016, pp. 93–98.

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

[Zhou et al.2017] Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1095– 1104. Association for Computational Linguistics

[See et al.2017] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the

Table 5: Rouge scores of generated summaries

	Rouge 1			Rouge 2			Rouge L		
Models	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
Epoch 5, batch=64, 2 layers, dropout=07	0.1122	0.1004	0.1051	0.04578	0.0420	0.04343	0.1105	0.099076	0.1035
Epoch 10, batch=32, 2 layers, dropout=0.7	0.1158	0.1020	0.1073	0.04725	0.04355	0.04485	0.11396	0.1006	0.10578
Epoch 10,batch=64, 3layers, dropout=0.7	0.1119	0.09974	0.1044	0.04655	0.04269	0.0440	0.11096	0.0990	0.1035
Epoch 10, batch=64, 3 layers, dropout=0.5	**0.13247**	**0.1185**	**0.12980**	0.05379	0.0497	0.0511	**0.130559**	**0.11715**	**0.1222**
Epoch 20, batch=64, 2 layers, dropout=0.7	0.118825	0.10990	0.113194	0.052630	0.04995	0.05083	0.11704	0.10851	0.111638
Epoch 20, batch=64, 3 layers, dropout=0.7	0.12734	0.119775	0.12222	0.055925	0.05390	0.054392	0.124920	0.117740	0.1200
Epoch 20, batch=64, 3 layers, dropout =0.5	**0.14541**	**0.1254**	**0.134665**	0.067456	0.06423	0.06580	**0.14235**	**0.1245**	**0.132827**

point: Summarization with pointergenerator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1073–

[Venugopalan et al.2015] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence - video to text. CoRR, abs/1505.00487.

[Vinyals et al.2015] O. Vinyals, M. Fortunato, and N. Jaitly. 2015. Pointer Networks. ArXiv e-prints, June.

[Paulus et al.2017] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," arXiv preprint arXiv:1705.04304, 2017.

[Mikolav et al.2013] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems. arXiv:1310.4546. Bibcode:2013arXiv1310.4546M.