Proceedings of the SMART–2019, IEEE Conference ID: 46866
8th International Conference on System Modeling & Advancement in Research Trends, 22nd–23rd November, 2019
College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, India

Generating Bengali News Headlines: An Attentive Approach with Sequence-to-Sequence Networks

Mushfiqus Salehin[1], Ashik Ahamed Aman Rafat[2], Fazle Rabby Khan[3] and Sheikh Abujar[4]
Dept. of Computer Science and Engineering,
Daffodil International University, Dhaka, Bangladesh
E-mail: [1]mushfique15-7056 @diu.edu.bd, [2]aman15-6858@diu.edu.bd,
[3]rabby15-6727 @diu.edu.bd, [4]sheikh.cse@diu.edu.bd

***Abstract*— This age of data-driven innovation has made automated relevant and important data extraction a necessity. Automated text summarization has made it possible to extract relevant information from large amounts of data without needing any supervision. But the extracted information could seem artificial at times and that's where the abstractive summarization method tries to mimic the human way of summarizing by creating coherent summaries using novel words and sentences. Due to the difficult nature of this method, before deep learning, there hasn't been much progress. So, during this work, we have proposed an attention mechanism-based sequence-to-sequence network to generate abstractive summaries of Bengali text. We have also built our own large Bengali news dataset and applied our model on it to show indeed deep sequence-to-sequence neural networks can achieve good performance summarizing Bengali texts.**

Keywords—abstractive text summarization, Bengali text summarization, LSTM, attention-based, headline generation

I. INTRODUCTION

Automatic text summarization has been among the important challenges for language related tasks since the beginning of the digital era [1]. As the amount of data in this digital age becomes insurmountable and data itself becomes more diverse, it is becoming a necessity to extract relevant and key information from the vast sea of data. Beside many other types of data, the number of documents is also increasing, so it's necessary for us humans to understand the essence of them with as little effort as possible. That's where automatic text summarization comes into play. A typical summarization task involves finding out the key information and represent it in a meaningful yet condensed way and automatic text summarization aims to automated this whole process, eliminating supervision during handling of large amount of data in a reliable way.

Automatic text summarization methods mainly evolve into two categories, 1) Extractive and 2) Abstractive text summarization. Extractive text summarization involves finding the key sentences of a document and extracting them to construct a summary. Many algorithms use a ranking [2] to find the key sentences of a document the arrange them into a summary. Some other approaches use graph representation [3] of sentences to find the important sentences. These extractive methods manage to find important sentences or topics pretty accurately but their generated summaries are not coherent. They lack the novel essence when compared to a human written summary instead they feel out of context or artificial.

When generating summaries, humans not only extract the information from the text but also restructures them differently than the source text so that they can contain much more information using less sentences and this is known as the abstractive way of summarization This way, humans often include novel words or sentence restructuring which is pleasant to read and feels more humane. Historically, generating abstractive summaries which mimics humans' traits has been a difficulty. That's why most of the earlier works on this topic covers the extractive ways of summarization.

Data has been the main driving force behind many artificial intelligence innovations recently and the rise of deep learning-based data driven methods only boosted it. With the help of deep neural models, it became easier to construct summaries rather than extracting it. Deep neural networks made using convolutional [17] or recurrent [18] architectures driven with large datasets [27][28] yielded highest scores on all benchmarks. This networks also showed they are scalable to even larger dataset and by improvements and fine tuning we can get even better results.

Bengali has been a low resource language when it comes to NLP related tasks or resources despite being the 8th most spoken language of the world. So, during this work, our first target is to collect a huge dataset of headline-article pairs to train a good deep neural network. Then we want to train our proposed model on that dataset to see how well our deep learning model performs on Bengali abstractive summarization.

In our model we'll use a bidirectional LSTM as our encoder along with two attention mechanisms. Then our decoder will be a simple LSTM which will use a beam search along with attention to generate predictions. Finally, we'll enforce parameters to restrict repletion of words in our decoder stage.

II. RELATED WORKS

Text summarization has been one of the key areas of application of NLP. NLP community has so far tackled all of extractive, compressive, abstractive summarization approaches. But more recently with the rise of deep learning,

the abstractive approach has gained more traction. In this section, we discuss the works that have been done so far for dominant languages like English and then we also discuss the limited number of works that concluded for our target language Bengali.

A. For High Resource Languages

LexRank [3] is one of the most recognized extractive summarization models. This model used intra-sentence cosine similarity to represent the graph of sentences which computes the importance of sentences, based on the eigenvector centrality concept. LexRank outperformed previous degree-based methods on various datasets. Since then many more approaches [2][4][5] have tried to improve extractive summarization including the most recent one based on RNNs by Nallapati et al [6].

There have been some notable works [7][8] on compression-based models also. Knight et al [9] proposed two separate models, one is based on noisy-channel and the second one is based on decision-tree model. For sentence compression and summary generation. Martins et al. [10] integrated extraction and compression into one global optimization problem. On a recent work Xu et al. [11] proposed similar model to combine extraction and compression but they did it with a neural approach.

Earlier approaches of generating abstractive summaries included Prior Knowledge based [13], Natural Language Generation (NLG) [14][15], Sentence Fusion [12] and Graph-Based [16] approach. Because generating abstractive summaries can be complex and difficult, none of the methods gained quite the result nor the attention towards them. But with the recent immergence of deep learning approaches and with the access to larger datasets, abstractive method of summarization is becoming a common thing.

In 2015, Rush et al. [17] proposed a novel method of summarization named Attention-Based Summarization (ABS) which incorporated deep learning methods into abstractive summarization. Their encoder was attention-based and the decoder incorporated a beam-search method to generate summaries after being trained on large corpora. They showed deep learning approaches are scalable to large amounts of data as data driven approaches becomes necessary because of the increasing amount of data around us.

As NLP problems involves handling sequence of data, Nallapati et al. [18] proposed RNNs with the application of sequence-to-sequence to tackle the text summarization problem. They used attention along with a switching generator-pointer to handle rare or unseen words. A similar method was used by the authors of the CopyNet [19] to incorporate copying of source words in case of unseen or rare words.

See et al. [20] uses a similar approach of copying source text using a pointer generator but uses a converge to avoid repletion of words which a lot of models above suffer from.

All of the above models usually do well on short text summarizations but a novel architecture based on deep reinforcement learning also does well on longer texts also fixing the repetition problem [21]. To attend to both input and also the output, they used an intra-attention method for each.

B. For Bengali Language

The number of works for Bengali language in the field of automatic text summarization has been relatively low. Most of the summarization work done for Bengali language involves the extractive approach [22][23] of summarization. Sarkar et al. [24] proposed a method which involved sentence ranking to generate summaries. It got a good recall score on a Bengali corpus.

The relative difficulty of producing abstractive summaries compared to extractive methods has kept the number of works to very low. Sunitha et al. [25] studied graph based abstractive summarization techniques and applied them on Bengali and other Indian languages.

Due to the heavy data and computational requirements, deep learning approaches to summarization for Bengali language have been extremely low. So, in this work we want to discover that dimly lit area.

III. DATA

The biggest challenge for this work for was getting the data for writing summaries. As Bengali is a low-resource language, there are no available public dataset to do summarization. So, we had to collect data on our own and make our own dataset of paragraph, summary pairs. In this section we are describing how we collected and then preprocessed our data for this task.

A. Data Collection

Our first challenge for our data collection was to choose the correct source of data. As we've seen majority of summarization tasks [17][18][20] have been performed on some kind news article datasets [26][27][28], we selected Bengali news portals for our source of data. As news portals often have their distinct writing patterns, we've selected three different news portals to eliminate any kind of biasing in our dataset towards certain writing pattern. Though we are not naming the portals we've used because we've used on the basis of educational purpose only.

The next challenge for us was to select a framework for scraping headline and article pairs from the news portals. We used Scrapy as our framework. Though Scrapy's internal downloader can download html content, it can't handle sites that relies on JavaScript to render its content. So, we had to use a different downloader created with the help of Selenium Webdriver Now we discuss our methodology to collect articles below:

1) Extracting article URLs from index: First we generate URL patterns of the article index pages. We used the archive sections of the portals as our index page. After generating the URLs, we get the HTML response of every index page and extract the links using Scrapy Selector's xpath commands.

2) Generating article requests: After getting the article links, we *yield* a request for the article link to the Scrapy engine. If the website renders its contents in JavaScript, we use *SeleniumRequest*. Scheduler schedules request for executing. After scheduler delivers it via downloader middleware.

3) Handling Requests Downloader middleware controls the requests that is to be sent to the downloader. If the request is a standard Scrapy one, it delivers it to the default

downloader. But if it's a *SeleniumRequest* it sends it to the Selenium webdriver for handling.

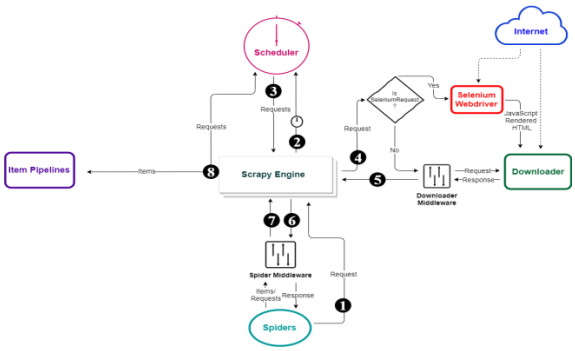

Fig. 1. Dataflow of our scraper

4) Rendering JavaScript content using Selenium webdriver: Selenium uses a headless Chrome instance to render the target webpage and waits until all the JavaScript contents are loaded. After rendering the JavaScript contents, it delivers them as HTML response to the standard Scrapy Downloader.

5) Processing the HTML response of articles: After the downloader sends the HTML response back to the Spiders, we again use xpath selectors to get the text data from the article and headline nodes.

6) Saving the collected data: We save the collected in JSON format using key value pairs.

B. Data Preprocessing

We have collected over 500,000 articles from the three news portals. Our next challenge was to clean and preprocess the data before applying them to our model. We followed below procedures to clean and preprocess our data:

- We removed erroneous data from our dataset by removing empty or incomplete article-headline pairs.
- We have replaced Bengali numbers such as ০, ১, ২, ৩, ৪, ৫, ৬, ৭, ৮, ৯ with the token *"#"* to eliminate context errors in our data.
- We placed each article and headline into a new line to separate them.
- After that, we divided our dataset into training and validation set, totaling four files because of separating headline and articles into separate files as well.
- We use CLTK tokenizer to tokenize the sentences and NLTK tokenizer to tokenize words in sentences loaded from both training and validation dataset.
- We use the token <*s*> to mark start and </*s*> to mark the end of a sentence.
- We created a dictionary to hold the most common words found in the whole dataset.

IV. Methodology

Our model is fairly similar to the neural machine translation model by Luong et al. [29]. We use beam search decoder for feed forward step and attention mechanisms based on Bahdanau et al [30] and Luong et al [29].

A. Embedding

In our model, we represent the input words of the article and summary as embeddings of the said words. We use embeddings because unlike one-hot vectors, embeddings carry meaningful information about the words. To generate the embeddings, we used Facebook's Fasttext word embedding model by Bojanowski et al. [31].

Fasttext model is unique in their way of taking sub word information when generating embeddings which is vital for a morphological language like Bengali. Fasttext's base model is similar to the Skip-gram model by Mikolav et al. [32]. According to the author, if we give a function token sequence, let $t_1, t_2, \ldots, t_R$, it'll try to increase "log-likelihood" as following:

$$\sum_{r=1}^{R} \sum_{i \in I_r} \log p(t_i|t_r) \quad (1)$$

Here, if we have context word t_i, the probability of getting it can be calculated using the vectors of the nearby words t_r.

In the sub word model of Fastext [31] each word consists of "*n-gram"* bag of characters. For example, if we take the word "ময়ূরপঙ্খী" (Peacock shaped boat) we get following *n-grams*:

<ময়ূ, ময়ূর, য়ূরপ, রপঙ্খী, পঙ্খী>

And we also include the source word as,

<ময়ূরপঙ্খী>

For this example, we are wrapping the source word with < and > tags and taking the size of *l-gram* as 3. This method helps us to differentiate the "ময়ূর" tri-gram from the word "ময়ূর" (Peacock). To represent a word *t*, we sum the vector representation, v_m, of said word using following:

$$F(t,i) = \sum_{m \in \eta_x} v_m^T u_i \quad (2)$$

B. Encoder

A standard RNN can take a sequence of inputs, let $t = (t_1, \ldots, t_l)$, in our case, and output a sequence, let $y=(y_1, \ldots, y_l)$, by computing a hidden sequence $h = (h_1, \ldots, h_l)$, at each time step *i*.

$$h_i = \alpha(W_{th} t_i + W_{hh} h_{i-1} + b_h) \quad (3)$$

$$y_i = W_{hy} h_i + b_y \quad (4)$$

But when it comes to remembering features of a long sequence, RNNs are weak due to the vanishing gradient problem. To generate summaries, we need to input long sequence of words as inputs, so a standard RNN won't perform well for this task. To solve this problem, we used Long Short-Term Memory **Error! Reference source not found.** as the architecture of our encoder. The LSTM architecture can be described using the following algorithm:

$$s_i = \sigma\left(W_{Fs} F_{t_{i-1}} + W_{hs} h_{i-1} + b_{Fs} + b_{hs}\right) \quad (5)$$

$$f_i = \sigma\left(W_{Ff} F_{t_{i-1}} + W_{hf} h_{i-1} + b_{Ff} + b_{hf}\right) \quad (6)$$

$$o_i = \sigma\left(W_{Fo}F_{t_{i-1}} + W_{ho}h_{i-1} + b_{Fo} + b_{ho}\right) \quad (7)$$

$$\hat{c}_i = \tanh\left(W_{Fq}F_{t_{i-1}} + W_{hq}h_{i-1} + b_{Fq} + b_{hq}\right) \quad (8)$$

$$c_i = f_i c_{i-1} + s_i \hat{c}_i \quad (9)$$

$$h_i = o_i \tanh(c_i) \quad (10)$$

Here, at each *i* time step LSTM outputs a hidden state which is denoted by h_i. If our current word is within our vocabulary *d* then we use the corresponding embedding matrix for that word pretrained using Fasttext. If the token is not in the embedding, then we generate a zero vector as the embedding for that token. For the end tokens <*s*> and </*s*>, we initialize their embeddings as random.

Our LSTM is a bidirectional LSTM so it has both a forward hidden state and as well as a backward hidden state. We concatenate both cell and hidden state's forward and backward states and send them to decoder and attention mechanism respectively.

$$h_i^{a,enc} = h_i^f \oplus h_i^b \quad (11)$$

$$c_i^{a,enc} = c_i^f \oplus c_i^b \quad (12)$$

C. Decoder

We have used a unidirectional basic LSTM decoder for our model. Our decoder takes the hidden states h^a and outputs a target which our case is a summary. Our decoder's initial hidden state and cell state is initiated as following:

$$h_0^{dec} = \tanh(W_{hh}h_i^a + b_{hh}) \quad (13)$$

$$c_0^{dec} = c_i^a \quad (14)$$

In every decoding time step *i*, we calculate the decoder hidden state h_i^{dec} using the previous input token and hidden state as:

$$h_i^{dec} = \text{LSTM}\left(h_{i-1}^{dec}, F_{t_{i-1}}\right) \quad (15)$$

We generate the target token *x* in from our vocabulary *d* using the following probability distribution:

$$P_{d_i} = \text{softmax}\left(W_{hP}h_{i-1}^{dec} + b_{hP}\right) \quad (16)$$

D. Attention

Different attention mechanisms have brought great results in various NLP tasks which require selective attention on certain parts of the text.This mechanism selects certain parts of article to focus in every decoder time step as well as takes final hidden and cell states as inputs. The attention mechanisms we used here was given by Bahdanau et al [30] and the other Luong et al [30]. We'll compare these two attention mechanisms performance in the result analysis section. We used weight normalization [34] along with the attention mechanism to speed up the training process.

1) Bahdanau Attention [30]: Bahdanau attention mechanism our context vector c_i is replaced by a attention context vector *b* by inputting hidden states from both encoder and decoder. This context vector b_i^{enc} decides which part of the article to pay attention to when decoder decides the next output summary token. The author defined this context vector b_i^{enc} using the following:

$$b_i^{enc} = \theta_{ik}^{enc} h_i^{a,enc} \quad (17)$$

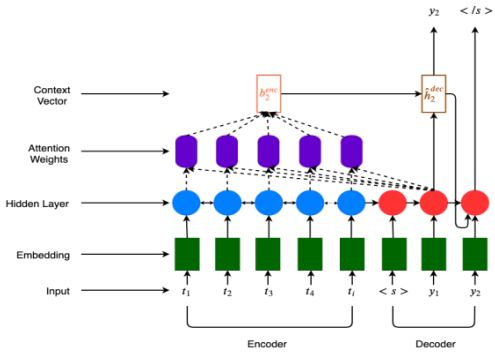

Fig. 2. Encoder-decoder model with attention

We can calculate θ_{ik}^{enc} using alignment scoring as follows:

$$\theta_{ik}^{enc} = \frac{\exp(e_{ik}^{enc})}{\sum_{n=1}^{K} \exp(e_{in}^{enc})} \quad (18)$$

The alignment score e_{ik}^{enc} is calculated using the following alignment function:

$$e_{ik}^{enc} = u^T \tanh(W_{h\theta}^{dec} h_{i-1}^{dec} + W_{h\theta}^{enc} h_k^{a,enc}) \quad (19)$$

.

2) Luong Attention [29]: Luong attention model is very similar to Bahdanau attention [30] model. It considers the current decoder step and all the encoder steps when computing the alignment score. It has three different methods to calculate alignment score e_{ik}^{enc}. They are:

a) Dot method: $(h_k^{enc})^T h_i^{dec}$

b) General method: $(h_k^{enc})^T W_\theta h_i^{dec}$

c) Concat method: $u_\theta^T \tanh(W_\theta[h_k^{enc}; h_i^{dec}] + b_\theta)$

It calculates alignment and context vector in a similar fashion as that (17), (18) of Bahdanau et al [30] .

E. Beam Search

Beam search incorporates behaviors of exact search but its also efficient like greedy search [17]. Beam search finds the top-K results from given a vocabulary, *d*. In our model, we use a beam search decoder. This enables a way for us to have the full vocabulary at hand yet working only with K-number of possibilities at a time. Though this introduces another problem for us, repetition. We fix this by forcing a parameter on beam search which restricts it from taking the same word multiple times for consideration.

V. Experiments

In this part, we discuss our experiments and outcomes of those experiments and how we set up our experiment environment and parameters to achieve those.

A. Environment Setup and Parameters

First, we prepared our dataset and trained our Fasttext embedding with both our training and validation set. We used

a embedding size of 300 and the vocabulary size is 42569. We trained our embedding for 30 epochs to get an optimal result.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Luong [29] attn.	33.60	15.64	31.41
Bahdanau[30] attn.	**39.88**	**18.62**	**37.15**

TABLE I. ROUGE SCORES

Fig. 3. Comparison between two proposed models on our own dataset

TABLE II. COMPARISON BETWEEN ENGLISH AND BENGALI MODELS

Model	R-1	R-2	R-L	Dataset	Language
Nallapati [18]*	35.46	13.30	32.65	CNN/D.M.	EN
See [20]*	39.53	17.28	36.38	CNN/D.M.	EN
Paulus [21]*	39.87	15.82	36.90	CNN/D.M.	EN
Luong attn.	33.60	15.64	31.41	500K B.N.	BN
Bahdanau attn.	**39.88**	**18.62**	**37.15**	500K B.N.	BN

*Not directly comparable

Fig. 4. Comparison between our two models with some of the state of the art models trained on CNN/Dailymail English news dataset

Article: জামালপুর সদর উপজেলায় দুই ছেলের লোহার শাবলের আঘাতে বাবার মৃত্যু হয়েছে। গতকাল শুক্রবার রাত #টার দিকে উপজেলার নরুন্দি এলাকার আড়ালিয়া গ্রামে এ ঘটনা ঘটে। ঘটনার পর থেকে দুই ছেলে পলাতক রয়েছেন।
Father of two sons died after being hit by an iron shawl in Jamalpur Sadar Upazilla. The incident took place at Aralia village in Narundi area of the upazila on Friday night #pm yesterday. Two boys have been absconding since the incident.
Gold: দুই ছেলের হাতে বাবা খুন!
Father killed in the hands of two sons!
Bahdanau: জামালপুরে শাবলের আঘাতে বাবা নিহত
Father killed in Shabal attack in Jamalpur
Luong: জামালপুরে শাবলের আঘাতে বাবার মৃত্যু
Father died in Shabal attack in Jamalpur

Fig. 5. Generated summaries of two models compared against the gold (reference) summary. Though both can be considered good summaries, they achieve a low score on ROUGE metrics.

To implement our model, we've used TensorFlow 1.14 framework. We've used two bidirectional LSTM hidden layers as our encoder each with hidden 150 cells. We've set a learning rate of 0.001 and beam width of 10. We trained our datasets as minibatches of size 64 for 20 epochs.

In the hardware side of things, we've used a machine with NVIDIA Tesla T4 GPU with 12.5GB memory to train

B. Comparisons

We've compared our two proposed models based on two different attention mechanisms to see which one performs better on Bengali language. We've calculated ROUGE 1, 2 and L scores [35] of both of our models which has been the standard scoring metrics for summarization tasks.

As there are no public datasets available for Bengali text summarization, we couldn't compare our results with the very few works that has been done on this topic. .

We've also stacked our results against some of the best models for English language to show how our model fares against the English language models. Though the results can't be compared directly as the datasets are different, but it gives us some intuition about our state against the stablished models.

C. Results

In Fig. 3, between our two models, the model based on Bahdanau attention [30] performs better than the Luong attention-based model [29]. It improves on all three metrics of the ROUGE score. The most notable improvement was the ROUGE-1 by +6.

In Fig. 4, we can see compared to some of the state-of-art models, our model performs on par if not better. Specially the one based on Bahdanau attention scores higher than the reinforced learning model introduced by Paulus et al [21].

In Fig. 5, we can see an example of generated summaries against the gold summary. Both of the models in this case output different but coherent summaries.

D. Analysis

In the results presented in both Fig. 3 and Fig. 4 shows our models performs in a good manner on our dataset and the score is similar to that of the state-of-the-art standards of other languages.

In Fig. 5, we see both of our models managed to learn the similarity of the words "জামালপুর"-"জামালপুরে" thanks to the sub word representation model of Fasttext. We can also see our models managed to get a good summary of the given article but yet got a low score of 19.95 in ROUGE-1. It shows ROUGE isn't a perfect way of scoring coherent abstractive summaries.

We also sometimes see in our results of the full validation set that sometimes our model can get facts wrong and sometimes it can predict incorrect words in case of rare/unseen words.

VI. CONCLUSION AND FUTURE WORKS

In our work, our proposed attention mechanism-based sequence-to-sequence model achieve good performance on a Bengali dataset. Despite having no reference works in Bengali, we achieve comparable scores against the well stablished models for other languages. Though it achieves a good ROUGE score, but sometimes it can get the factual information wrong. In a future work we hope to represent an improved model which will perform good with unseen words and produce correct factual information.

REFERENCES

[1] Luhn, Hans Peter. "The automatic creation of literature abstracts." *IBM Journal of research and development* 2, no. 2 (1958): 159-165.

[2] Mihalcea, Rada, and Paul Tarau. "Textrank: Bringing order into text." In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404-411. 2004.

[3] Erkan, Günes, and Dragomir R. Radev. "Lexrank: Graph-based lexical centrality as salience in text summarization." *Journal of artificial intelligence research* 22 (2004): 457-479.

[4] Wan, Xiaojun, Jianwu Yang, and Jianguo Xiao. "Manifold-Ranking Based Topic-Focused Multi-Document Summarization." In *IJCAI*, vol. 7, pp. 2903-2908. 2007.

[5] Murray, Gabriel, Steve Renals, and Jean Carletta. "Extractive summarization of meeting recordings." (2005).

[6] Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou. "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents." In *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[7] Wang, Lu, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. "A sentence compression based framework to query-focused multi-document summarization." *arXiv preprint arXiv:1606.07548* (2016).

[8] Zajic, David, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. "Multi-candidate reduction: Sentence compression as a tool for document summarization tasks." *Information Processing & Management* 43, no. 6 (2007): 1549-1570.

[9] Knight, Kevin, and Daniel Marcu. "Summarization beyond sentence extraction: A probabilistic approach to sentence compression." *Artificial Intelligence* 139, no. 1 (2002): 91-107.

[10] Martins, André FT, and Noah A. Smith. "Summarization with a joint model for sentence extraction and compression." In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pp. 1-9. Association for Computational Linguistics, 2009.

[11] Xu, Jiacheng, and Greg Durrett. "Neural Extractive Text Summarization with Syntactic Compression." *arXiv preprint arXiv:1902.00863* (2019).

[12] Barzilay, Regina, and Kathleen R. McKeown. "Sentence fusion for multidocument news summarization." *Computational Linguistics* 31, no. 3 (2005): 297-328.

[13] Radev, Dragomir R., and Kathleen R. McKeown. "Generating natural language summaries from multiple on-line sources." *Computational Linguistics* 24, no. 3 (1998): 470-500.

[14] Saggion, Horacio, and Guy Lapalme. "Generating indicative-informative summaries with sumUM." *Computational linguistics* 28, no. 4 (2002): 497-526.

[15] Jing, Hongyan, and Kathleen R. McKeown. "Cut and paste based text summarization." In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pp. 178-185. Association for Computational Linguistics, 2000.

[16] Ganesan, Kavita, ChengXiang Zhai, and Jiawei Han. "Opinosis: A graph based approach to abstractive summarization of highly redundant opinions." In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pp. 340-348. 2010.

[17] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." *arXiv preprint arXiv:1509.00685* (2015).

[18] Nallapati, Ramesh, Bowen Zhou, Caglar Gulcehre, and Bing Xiang. "Abstractive text summarization using sequence-to-sequence rnns and beyond." *arXiv preprint arXiv:1602.06023* (2016).

[19] Gu, Jiatao, Zhengdong Lu, Hang Li, and Victor OK Li. "Incorporating copying mechanism in sequence-to-sequence learning." *arXiv preprint arXiv:1603.06393* (2016).

[20] See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." *arXiv preprint arXiv:1704.04368* (2017).

[21] Paulus, Romain, Caiming Xiong, and Richard Socher. "A deep reinforced model for abstractive summarization." *arXiv preprint arXiv:1705.04304* (2017).

[22] Das, Amitava, and Sivaji Bandyopadhyay. "Opinion summarization in Bengali: a theme network model." In *2010 IEEE Second International Conference on Social Computing*, pp. 675-682. IEEE, 2010.

[23] Haque, Md Majharul, Suraiya Pervin, and Zerina Begum. "Automatic Bengali news documents summarization by introducing sentence frequency and clustering." In *2015 18th International Conference on Computer and Information Technology (ICCIT)*, pp. 156-160. IEEE, 2015.

[24] Sarkar, Kamal. "Bengali text summarization by sentence extraction." *arXiv preprint arXiv:1201.2240* (2012).

[25] Sunitha, C., A. Jaya, and Amal Ganesh. "A study on abstractive summarization techniques in indian languages." *Procedia Computer Science* 87 (2016): 25-31.

[26] Harman, Donna, and Paul Over. "The effects of human variation in duc summarization evaluation." In *Text Summarization Branches Out*, pp. 10-17. 2004.

[27] Napoles, Courtney, Matthew Gormley, and Benjamin Van Durme. "Annotated gigaword." In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pp. 95-100. Association for Computational Linguistics, 2012.

[28] Hermann, Karl Moritz, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. "Teaching machines to read and comprehend." In *Advances in neural information processing systems*, pp. 1693-1701. 2015.

[29] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).

[30] Bahdanau, Dzmitry, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. "End-to-end attention-based large vocabulary speech recognition." In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4945-4949. IEEE, 2016.

[31] Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information." *Transactions of the Association for Computational Linguistics* 5 (2017): 135-146.

[32] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111-3119. 2013.

[33] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9, no. 8 (1997): 1735-1780.

[34] Salimans, Tim, and Durk P. Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." In *Advances in Neural Information Processing Systems*, pp. 901-909. 2016.

[35] Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." In *Text summarization branches out*, pp. 74-81. 2004.

