2019 22nd International Conference on Computer and Information Technology (ICCIT), 18-20 December, 2019

A Machine Learning Approach to Extract Keyphrases from Bengali Document using CNN-Bidirectional LSTM

Nishat Tasnim Ahmed Meem
Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
meemnishat@gmail.com

Muhammad Mahir Hasan Chowdhury
Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
mahirhasancse@gmail.com

Md. Mahfuzur Rahman
Computer Science and Engineering
Shahjalal University of Science and Technology
Sylhet, Bangladesh
mahfuzsustbd@gmail.com

***Abstract*—Keyphrases are single or multiple word phrases of a document which describe the principal topics of that document. These keyphrases help readers to get an overview of the document. In this paper, we proposed a system that uses the combination of Convolutional Neural Network and Bidirectional Long Short-Term Memory (BiLSTM) Recurrent Neural Network (RNN) to automatically detect keyphrases from a document. We also used some preprocessing steps to clean and generate candidates keyphrases to train the model. Convolutional Neural Network can analyze semantic meanings of sentences. Bidirectional LSTM can learn the relations among words in the sentences. A Bengali pre-trained word embedding is used in this work.**

***Keywords*—Keyphrase, Keyphrase Extraction, Keywords, BiLSTM, RNN, CNN, Convolutional Neural Network, Word Embedding, FastText, Neural Network**

I. INTRODUCTION

Keywords or Keyphrases are the significant words in document or text those carry the concrete idea about that document or text. Automated Keyphrase Extraction (KPE) is defined as a task that automatically identifies a set of words without any help of a human reader that best describes the subject of a document. Keyphrases serve various goals such as summarization, categorization, indexing, precise searching in the search engine, data mining etc. Along with any other languages, contents in Bangla over the net is increasing in tremendous number day by day. For processing or mining, this vast amount of text or article, an automated keyphrase extraction system is much demanded. Hence an efficient automated keyphrase extraction system for Bengali language is a must needed tool. In this paper, we proposed a method that uses both CNN and Bidirectional LSTM recurrent neural network for an automated keyphrase extraction system. We used several pre-processing steps like tokenization, stop words removal, verb suffix elimination, stemming etc. in both approaches. Convolutional Neural Networks(CNN) are designed to learn efficiently the features of a specific concern regardless of the locality. Bidirectional LSTM which is a Recurrent Neural Network(RNN) generates the output based on both previous and future elements of a sequence. We compared this model with regular CNN and Bidirectional LSTM networks and also with our previous work [1]. In section 2, we present the related work. We discuss some background topics in section 3. In section 4, the proposed methodology has been discussed. We present the evaluation and the experimental results in section 5.

II. RELATED WORKS

We studied some papers on this topic to have deep knowledge about works done on this topic.

Basaldella et al. [2] presented a method based on Bidirectional Long Short-Term Memory Recurrent Neural Network. Bidirectional LSTM works with both previous and future context of the words and does not requires features to rank the words. They used Stanford's GLOVE embedding which is a pre-trained word embedding for their work. This method performed better than competitive methods.

Mahata et al. [3] presented an unsupervised technique named *Key2Vec*. Specifically, they used phrase embeddings for thematic representation for scientific articles. They used *FastText* as their embedding technique rather than *Glove* or *Word2Vec* because it captures both semantic and morphological relations between words. After generating candidate keyphrases they used *PageRank Algorithm* to rank them. Zhang et al. [4] worked for extracting keyphrases from microblog using conversion context. They used four types of neural encoders - averaged embedding, RNN, attention, and memory network for representing conversation context. Their framework consists of two parts - a keyphrase tagger and a conversation encoder. In [5], Zhang et al. proposed an approach for extracting keyphrases from tweets of Twitter which has length limitations. They used deep recurrent neural network (RNN) model to combine keywords and context information. Meng et al. [6] proposed a method named *Deep Keyphrase Generation*. This method captures the deep semantic meaning of the content to extract not only the keyphrases that appear in the text but also

978-1-7281-5842-6/19/$31.00

the absent keyphrases based on the semantic meaning of the text. They used an encoder-decoder framework for this.
Sujata et al. [7] addressed keyphrase extraction as a *sequential labeling* task. They used a basic set of features along with predictions from an unsupervised model as a feature.

In our previous work on this topic is [1], we proposed a system that uses Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) and also implemented Multilayer Perceptron (MLP) network to compare the performance of our proposed LSTM approach. We applied several pre-processing steps on a document to generate the candidate keyphrases. Finally, we found better performance in our proposed approach with compared to the MLP network.

III. BACKGROUND

A. Bidirectional LSTM

LSTM is a special kind of Recurrent Neural Network (RNN). Generally, RNN has internal memory and it can remember important input sequence using this memory. LSTM or Long short-term memory has a forget gate so that it can forget certain input sequence when needed. This helps in avoiding the vanishing gradient problem. A major issue with LSTM is they learn only from previous inputs. Sometimes it may be needed to consider both previous and future inputs. For example, "সে বলল, বেলি আমার প্রিয় ফুল" and "সে বলল, বেলি খুব ভাল একটি মেয়ে". In these two sentences if we look at the word "বেলি" and the previous two words সে বলল we might not be able to understand if the word "বেলি" refers to a person or flower. To resolve this ambiguity we need to consider future words of the sequence.
Bidirectional LSTM is putting two LSTM together. One to learn the input sequence in previous order and another to learn in forwarding timestamps. According to M Schuster et al [8], "To overcome the limitations of a regular RNN we propose a bidirectional recurrent neural network (BRNN) that can be trained using all available input information in the past and future of a specific time frame. The idea is to split the state neurons of a regular RNN in a part that is responsible for the positive time direction (forward states) and a part for the negative time direction (backward states)."

B. Convolutional Neural Network

One of the special kinds of deep neural networks is Convolutional Neural Network. It is specialised in various pattern recognition problems. Convolutional Neural Network uses multiple identical copies of the same neuron. So, it can learn a neuron and use it for learning a model and also reducing error. Convolutional Neural Network can be one or multidimensional. Samples at different points in time are collected which are equally spaced. To notice all local properties of the samples, a group of neurons can be created to compute certain features at small time segment. Finally, the output of this layer is fed into a fully connected layer. This is basically single layer convolutional network. Since the convolutional layer has a composable property, multiple layers of convolution can put on top of other layers. There is a layer called max-pooling layer which can be used in CNN to work on a large section of the data. The layer collects maximum features over the small block of the previous layer. So, the output can tell us if a feature is present in the previous layer or not. Fig. 1 shows the structure of a multilayer CNN with max-pooling layer where **A** and **B** are convolutional layers and X_0, X_1,....., X_8 are inputs.

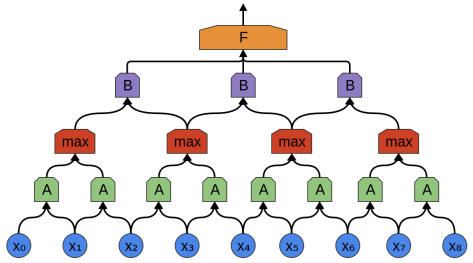

Figure 1. Structure of 2-Layer CNN with Max-Pooling Layer [9]

IV. METHODOLOGY

In this section, we briefly describe the proposed approach for keyphrase extraction. We'll use punctuation removal, stop words filtering, sequence generation for preprocessing Bengali documents. Then we will apply two deep machine learning approach. These are:

1) Bidirectional LSTM
2) Convolutional Neural Network

Bidirectional LSTM and CNN are better approaches for our work as we have discussed in the earlier section. We have implemented this network with and without pre-trained word embeddings to see its effect in for keyphrase extraction.

A. Datasets

For Bengali datasets, we have crawled German international broadcaster [1] online Bangla edition. We saved the documents in JSON(Javascript Object Notation) format. The reason behind collecting datasets from the German International broadcaster is that the article has author-assigned keyphrases which will be helpful for the training process of the neural net. The average number of keyphrases assigned by the author is 9 per article and the average number of sentences per document is 32.

B. Pre-processing

In the pre-processing stage, our goal is to clean each of the articles and tokenize them into sentences, furthermore removing stop words from the sentences and generating sequences.

1) Punctuation Removal: We used the regular expression in Fig. 2 to remove the frequently occurred punctuation marks in a document. Next, we eliminate stop words from the document.

2) Stop Word Elimination: Stop word removal is very necessary for keyphrase extraction because these words appear too often and create noise in the dataset. After removing

[1] http://www.dw.com/bn/

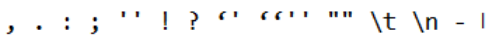

Figure 2. Regular Expression

punctuation, we eliminate stop word from the text. We use a stop word list containing 393 stop word collected from [11]. Stop words are removed based on the exact match. Some examples of stop words are shown in Fig. 4. After pre-processing of the documents, these documents are fed into our model as inputs.

3) Sequence Generation: After removing stop words from the text, we generated a sequence from the remaining text. This sequence was fed into our model as outputs. We denote non-keyphrase word as **1**, the word in the starting of a keyphrase as **2** and other words in a keyphrase word as **3**. Then we generated a sequence of number from the documents. The reason behind tagging the words like this is a keyphrase can one or more words long. When we will be predicting keyphrases with our model from a document, 2 will denote that this word is a keyphrase. If we get a 3 for a word after getting a 2 for the previous word, then these two words will be together in a keyphrase. Again, If we get a 2 for a word after getting a 2 for the previous word, then these two words will be two different keyphrases. So this tagging method will help us to detect the beginning and the end of a keyphrase. Fig. 3 shows how a document is changed after applying all pre-processing steps.

C. Pre-trained Word Embedding

Word embeddings are a natural language processing technique. It maps words' semantic meaning in a geometric space. Every word in the embedding is given a numeric vector. The distance between two word's vector is the semantic relation between the two words. For example, 'খেলা' and 'আকাশ' are semantically two different words so the distance between these two word's vector will be large. Again, 'খেলা' and 'ক্রিকেট' are related words so these should be embedded closer. There are different word embedding techniques like Word2vec, Glove, FastText, Gensim etc. It is quite a troublesome work to create and use your own word embedding. We have to download huge data, pre-process them, learn them for a long time etc. So it is convenient to use a pre-trained word embeddings. There are some pre-trained word embedding for Bangla that can be found on the Internet.

We have used a pre-trained word embedding distributed by *Facebook* [12]. As per their documentation [13], this pre-trained word vectors were distributed for 157 languages, trained on Common Crawl and Wikipedia using fastText. These models were trained using CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives. We have tested our model with or without this to observe its effect.

D. Designing the model

BiLSTM-CNN model has initial BiLSTM layers which take the word embedding as input. The output of this layer is then fed into CNN layers as input. CNN layer will extract the local features. We split our dataset into 75:15:15 for training, testing and validation. We have used *Keras* with *Tensorflow* back-end to implement this approach. After trying and testing different parameters, we have used these parameters for our final approach -

Table I
Parameters for our model

Embedding Dimension	300
Kernel Size	3
Batch Size	128
Filters	32
Dropout	0.4
LSTM Cell	128

We have used dropout in between the layers to avoid over-fitting. Adam optimizer was used to optimize the gradient descent. Table I shows the configuration of the model.

V. RESULTS AND DISCUSSION

A. Experiment result and comparison

We used Kaggle's[2] website which is a cloud computing environment that enables reproducible and collaborative analysis. They provide free GPU for the use which is NVIDIA Tesla K80. Training process took significantly less time using this. Per epoch took 60-70 second. We used Keras's early stopping rule to halt the training process if the loss does not reduce for 4 consecutive epochs. The loss of training process was plotted to monitor if over-fitting is happening. We can see in Fig. 5, that the loss is quite low indicating model is not over-fitting.

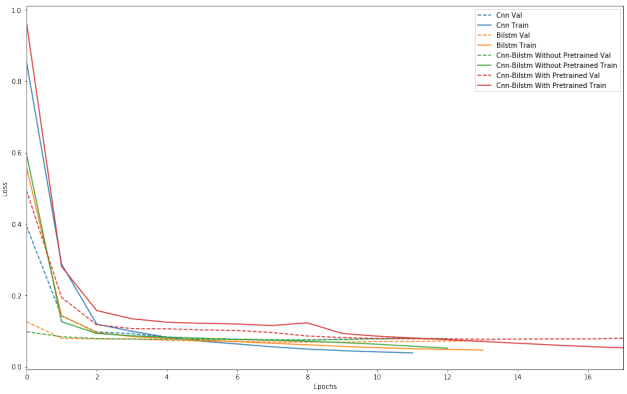

Figure 5. Training and Validation Loss of Different training processes

[2]http://www.kaggle.com/

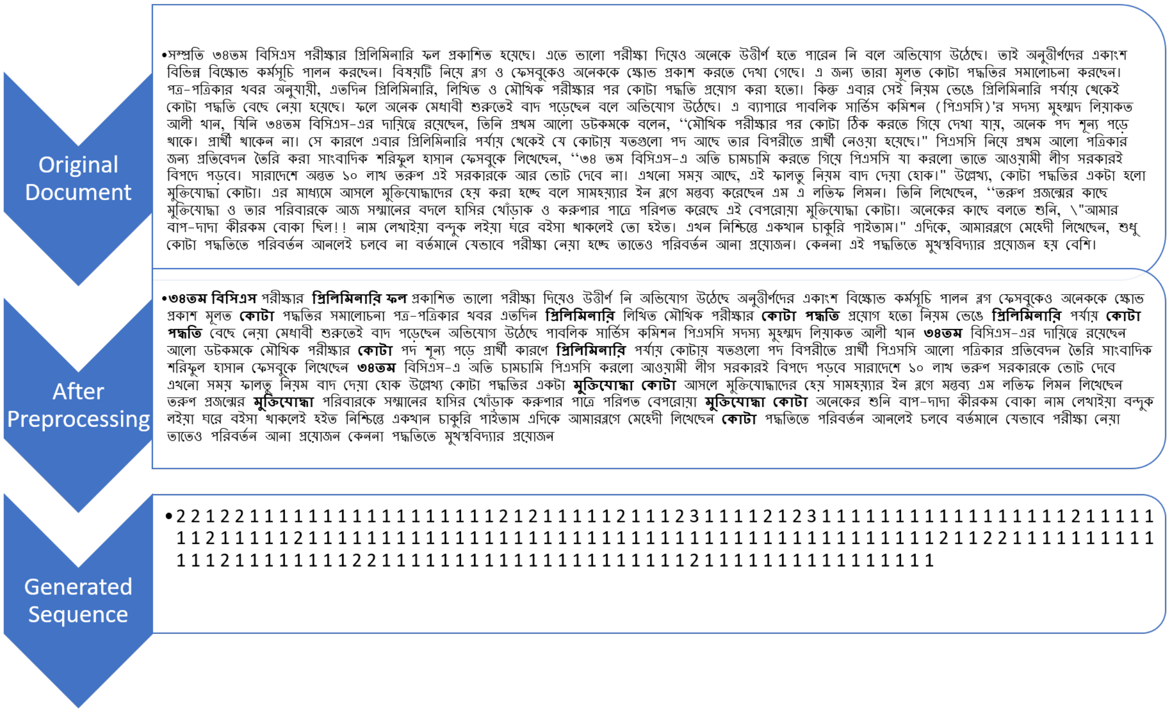

Figure 3. Processing a document

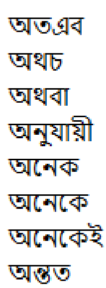

Figure 4. Stop Word List

Table II
Epoch of different approach

Approach	Epoch
CNN - BiLSTM with Pre-trained	17
CNN - BiLSTM without Pre-trained	11
BiLSTM	10
CNN	9

Different models took different number of epochs to complete the training process. Some model would learn quickly and start over-fitting quickly. Table II shows number of epochs each model took. Standard Bidirectional LSTM and CNN model took less epoch to complete the training. Their structure is comparatively simple and has less layers than CNN-Bidirectional LSTM. This can be the reason behind this.

Table III shows the performance of different implemented approach. It is clearly shown that CNN-Bidirectional LSTM approach outperforms the other standard approaches. So our intuition was correct. CNN was able to detect the patterns and Bidirectional LSTM Layer was able to detect the ordering. 1.5% improvement is also seen by using pre-trained Word embeddings. We used this best-performed model to predict keyphrases from a document. Figure 7 shows the result. 5 out of 7 keyphrases match the assigned keyphrases.

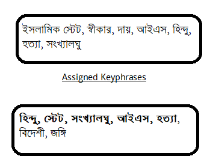

Figure 6. Detected keyphrases from a document

Table III
Performance of different approach

Model	Accuracy
CNN-BiLSTM with Pre-trained Embedding	95.9%
CNN-BiLSTM without Pre-trained Embedding	94.4%
BiLSTM	95.1%
CNN	94.5%

VI. Comparison with previous work

In our previous work [1], we proposed a system that uses Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) to detect keyphrases from a document. We generated unigram, bigram and trigram for the dataset and applied several pre-processing steps like - verb suffix elimination, stemming, stop word elimination etc. Accuracy of this model was 89.8% which is less than our current work. Pre-trained word vectors was not used in previous approach. Figure 7 shows the comparison of both.

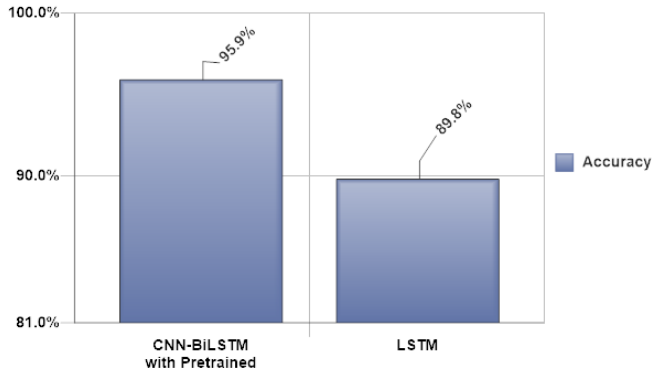

Figure 7. Comparison with previous work

A. Discussion

Most of the keyphrases in our dataset contain one word. Very few contain two or three words. As we have discussed in the earlier section that we have classified each word into 3 classes - non-keyphrase word as **1**, the word in the starting of a keyphrase as **2** and other words in a keyphrase word as **3**. So there will be not much words tagged as 3 comparing to the tagged word as 1 or 2 because not many word contains two or three words. Because of the limited number of example of 3, the model may not be able to detect them properly.

Introducing more preprocessing techniques can make the dataset more clean and noise-free. Most of the word in the dataset are less likely to be in their root word from. If a document's keyphrase is 'বাংলাদেশ', this word can be different forms like - 'বাংলাদেশে', 'বাংলাদেশের' etc. So these will not be properly tagged as keyphrases. To solve this problem, 'Stemming' process can be used.

VII. CONCLUSIONS

In our proposed system we introduced the combined CNN and bidirectional LSTM for generating keyphrases from a document. We cleaned our text to remove punctuation and then removed stop words. Then we tagged each word to generate an integer sequence to feed the models. The model contains a combination of BiLSTM and CNN layers. A pre-trained word embedding is also used in this approach. This approach shows an improvement over our previous approach. A good stemmer will play a vital role in improving the accuracy of the model. In this approach we processed each document and generated a number sequence(includes 0, 1 and 2) for each document as output. This problem can be defined as sequence-to-sequence prediction - predicting a number sequence for a given word sequence. An encoder-decoder model which consists of LSTM cells can work well with sequence prediction. So this model can be applied to see the performance of detecting keyphrases from documents.

References

[1] N. T. A. Meem, M. M. H. Chowdhury, and M. M. Rahman, "Keyphrase extraction from bengali document using lstm recurrent neural network," in 2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT), Sep. 2018, pp. 461–466.

[2] M. Basaldella, E. Antolli, G. Serra, and C. Tasso, "Bidirectional lstm recurrent neural network for keyphrase extraction," in Italian Research Conference on Digital Libraries. Springer, 2018, pp. 180–187.

[3] D. Mahata, J. Kuriakose, R. R. Shah, and R. Zimmermann, "Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), vol. 2, 2018, pp. 634–639.

[4] Y. Zhang, J. Li, Y. Song, and C. Zhang, "Encoding conversation context for neural keyphrase extraction from microblog posts," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), vol. 1, 2018, pp. 1676–1686.

[5] Q. Zhang, Y. Wang, Y. Gong, and X. Huang, "Keyphrase extraction using deep recurrent neural networks on twitter," in EMNLP, 2016.

[6] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," in ACL, 2017.

[7] S. D. Gollapalli and X. Li, "Keyphrase extraction using sequential labeling," CoRR, vol. abs/1608.00329, 2016.

[8] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673–2681, 1997.

[9] "Conv nets: A modular perspective," Available: http://colah.github.io/posts/2014-07-Conv-Nets-Modular/.

[10] C. Florescu and W. Jin, "Learning feature representations for keyphrase extraction," CoRR, vol. abs/1801.01768, 2018.

[11] "List of bangla stop words," Available: https://github.com/stopwords-iso/stopwords-bn, last accessed 5 September 2018.

[12] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[13] "Word vectors for 157 languages," Available: https://fasttext.cc/docs/en/crawl-vectors.html, last accessed 14 November 2019.