Deep Learning based Abstractive Summarization for English Language

Sangita Singh*, Jyoti Prakash Singh and Akshay Deepak

National Institute of Technology Patna, Bihar (India)

Abstract

Text summarization is one of the well-known issues in deep learning (DL) and natural language processing (NLP) in recent years. A sequence-to-sequence attention model based on recurrent neural networks has shown promising results for abstractive text summarization. Our main goal is to produce an abstractive summary of a text document that is succinct, fluid, and stable. In this regard, we have used the Indian Language Summarization (ILSUM)-2022 datasets, which are available on the Forum for Information Retrieval Evaluation (FIRE). We have used article text descriptions as our input data and generated a simple summary of that article description as an output. To assist in producing some extensive summaries, we have used bi-LSTMs in the encoding layer and LSTMs in the decoding layer. To create a concise summary of the thorough description, we applied the sequence to the sequence model. Our main goal was to increase the efficiency and reduce train loss of the sequence-to-sequence model to make a better abstractive text summarizer. In our experiment, we successfully reduced the training loss to 0.036 and demonstrated that our abstractive text summarizer can generate a short summary of English language.

Keywords

Deep Learning, Bi-LSTM, Sequence-to-Sequence, Encoder-Decoder model, NLP, ILSUM-2022 Datasets

1. Introduction

Now days, textual content like news articles, novels, legal documents, scientific papers, etc. is available in abundant quantity, and it is growing rapidly day by day. Because of the large amount of available information, extracting the required information takes a long time, and the extracted results are not always as precise as the required information. In this case, automatic text summarization (ATS) can then assist us in locating the relevant content. ATS is one of the most difficult tasks in natural language processing (NLP) and artificial intelligence (AI).

There are two approaches to ATS tasks: (i) an abstractive approach and (ii) an extractive approach. In an extractive approach, it pulls out the key phrases and words from the original text, separates out some crucial component, and then put everything together to create a summary. In an abstractive approach, new sentences that do not belong in the original document are generated. The NLP research community has paid surprisingly close attention to automatic text summarization for Indian languages. While large-scale datasets exist for languages including English, Chinese, French, and German, none exist for Indian languages. The vast majority of

Forum for Information Retrieval Evaluation, December 9-13, 2022, India

*Corresponding author.

sangitas.ph21.cs@nitp.ac.in (S. Singh); jps@nitp.ac.in (J. P. Singh); akshayd@nitp.ac.in (A. Deepak)

0000-0002-9318-2186 (S. Singh); 0000-0001-6854-8599 (A. Deepak)

existing datasets are either unavailable to the public or are too small to be useful. So, Forum for Information Retrieval Evaluation (FIRE) [2] will bridge the existing gap by creating reusable corpora for Indian language summarization through this collaborative project. They provide datasets in two major Indian languages for this task: Hindi and Gujarati. They also include an Indian English dataset, a widely recognized dialect that can differ significantly from English spoken elsewhere.

FIRE hosted a competition task for Indian Language Summarization (ILSUM) 2022, in which the dataset consists of articles and headline pairs from several of the country's most prominent newspapers. Each language receives around ≈10,000 in news articles. The task for each article is to write a meaningful fixed-length summary, either extractive or abstractive. In this regard, we examine all three Indian languages (Hindi, English, and Gujarati) provided by the organisers of ILSUM-2022 [3]. For this task, we develop a deep learning-based encoder-decoder approach for each dataset separately. To generate an extensive summary, we implemented the sequence-to-sequence model with one layer of bidirectional long short-term memory (Bi-LSTM) on the input text and one layer of unidirectional long short-term memory (LSTM) using global attention on the target text. An encoder learns the features of the input text (articles) and generates a fixed-length vector, which is passed to the decoder to learn and generate a summary that is relevant to the article.

The remaining of the paper is formatted as follows. Section 2 provides a synopsis of the related works. Section 3 presents our proposed framework for ILSUM-2022. Section 4 presents the proposed systems discovery and analysis of the results. Finally, in Section 5, we conclude the paper.

2. Related work

A neural machine translator is not the same as a traditional machine translator, but it is a vast approach to machine translator that has recently been developed Kalchbrenne et al. [4]. An individual neural network that can perform joint translation, usually with encoder and decoder. To improve the performance of the basic encoder and decoder, a fixed length of text is used as input and the decoder generates output in Bahdanau et al 2014 [5]. Abstractive text summarization generates a summary of a text document based on its intrinsic characteristics and selects the key content of the text document based on potential vocabulary. A summary of a text document is a sequence of actual words as input text in a source text document and predicted actual words of a sequence. For text summarization task, encoder decoder based RNN attention model on machine translation has been established by Ramesh Nallapatti et al 2016 [6]. Word topic distribution of LDA has been combined to the sequence-to-sequence model to improve abstractive sentence [7]. They proposed a heuristic method that used the LDA techniques to identify the optimal number of independent topics. Furthermore, for summarization, the two-tiered topic model based on the pachinko allocation model (PAM) is combined with the TextRank method [8]. To capture the sparse candidate topics under that low-rank matrix factorization model, a novel neighbourhood preserving semantic (NPS) measure was introduced. These techniques used the topic model as an additional mechanism to improve text generation. Nonetheless, these models have some sparseness issues and are difficult to train [9]. From

another aspect text summarization is further classified into extractive and abstractive models. Take into account the target user's lack of background knowledge or reading ability, and propose a linear combination of feature scores for social networks. Current text summarization research focuses primarily on word embedding, which represents each element in some way[10],[11]. However, word embedding does not completely solve the polysemy problem. To address this issue, Embeddings from Language Models (ELMOs)[12] used bidirectional LSTM to train the language model, whereas hierarchical LSTM can grasp different levels of granularity information.

3. Proposed Model

In this section, we discussed the methodology and datasets. We proposed an encoder-decoder-based deep learning model for abstractive summarization. The encoder layer consists of a BI-LSTM layer, which gives fixed-length features to the attention layer (as a value), and the decoder layer. Then the decoder layer extracts the relative information, which helps in generating better summaries. The output of the decoder layer is passed to the query vector and concatenation layers. So, the proposed model generates multi-sentence summaries, and the overall architecture is shown in Figure 1.

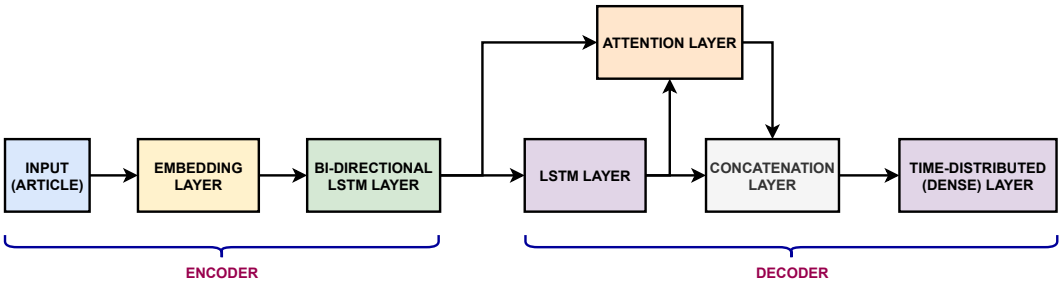


Figure 1: Proposed architecture diagram for the abstractive summarization.

3.1. Problem Assertion

We have large text corpora of articles and task is to create a relevant short summary (descriptors) that represents the corpora. Consider the following: the input sequence of article with D words $x_1, ...x_D$ comes from the vocabulary size V, and the generated output sequence is $y_1, ...y_s$ similar to article in meaning, $S < D$ – indicating that a summary sequence (S) is less than the article sequence (D) from the same vocabulary.

3.2. Data Collection

To achieve better results, deep learning algorithms require a large text corpus to learn discriminative features, as in our case. We used the FIRE-2022, which provides ILSUM-2022 datasets [3] in this work. By developing reusable corpora for Indian language summarization, they hope to

fill the current gap through this joint effort. They cover two important Indian languages in the first edition: Hindi and Gujarati, with over 350 million and 50 million speakers, respectively. They also contain Indian English, a well-known dialect that can differ significantly from English used elsewhere. Several of the nation's top newspapers' articles and headline pairs were used to build the dataset for this task. For each language, they provide 10,000 news articles. The dataset description is shown in Table 1. The goal is to produce an insightful fixed-length summary for each article that is either extractive or abstractive.

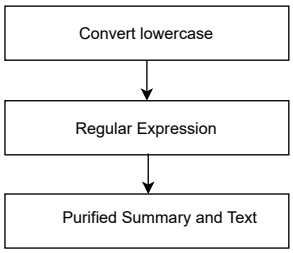
Table 1
Dataset description for Hindi, English, and Gujrati language

Datasets	Train data	Validation data	Test data
English	12565	898	4487
Hindi	7957	569	2842
Gujarati	8457	605	3020

3.3. Data Preprocessing

We followed two steps during the data preprocessing phase, as shown in Figure 2. First, we transform the entire text corpus to lowercase. Second, superfluous elements such as extra symbols {"@", " ∼ ", ",", "/", "$"} need to be eliminated from the text corpus. These superfluous elements can be eliminated using regular expression matching. After preprocessing, a cleaned text corpus has been obtained.

Figure 2: Different preprocessing steps.



- **Convert lowercase**:- Changing a word from uppercase to lowercase (NLP to nlp). Although words like "Book" and "book" have the same meaning when written in lower case, the vector space model represents them as two distinct words (resulting in more dimensions).
- **Regular Expression**:- Regular expressions, also known as RegEx, are string of characters primarily used to locate or replace patterns in text.

3.4. Word embedding

The importance of words depends on both their frequency and their similarity. We tested different pre-trained word-to-vector files like Glove, ConceptNetNumberbatch, FastText, and Word2vec to improve our model. In our case, Word2Vec outperformed the others. So, we have used this pre-trained model for further tasks. The total coverage of rare words in the article corpus is 1.88%, which is calculated by words having frequency 4 divided by the sum of total frequencies in the corpus, and the total percentage of rare words in the article corpus is 63.63%, which is calculated by the number of words having frequency 4 divided by the sum of total words in the corpus. As a result, we eliminate words that appear less than four times in the text article (frequency of words less than four). Whereas, in the summary, the total coverage of rare words in the vocabulary is 8.26%, and the total percentage of rare words in the vocabulary is 77.37%; based upon this, we eliminate words that appear less than six times (a frequency of less than six) throughout the summary. The above procedures are done after preprocessing steps. These types of words are eliminated to maintain the importance of the words and the coverage of these final words (vocabulary) in the article and summary.

3.5. RNN Encoder-Decoder

RNN has demonstrated promising results in tasks involving sequence processing, particularly when dealing with variable-length sequences[13]. Additionally, RNN based LSTM performs better in many tasks and is easier to train than vanilla RNN. As a result, we use an LSTM-based RNN as encoder to produce coarse encoding[14]. The LSTM can adaptively and captively capture dependencies of different time scales, which are defined as the following equations.

$$
\begin{aligned}
f_t &= \sigma_g(W_f.x_t + U_f.h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i.x_t + U_i.h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o.x_t + U_o.h_{t-1} + b_o) \\
c_t^{'} &= \sigma_c(W_c.x_t + U_c.h_{t-1} + b_c) \\
c_t &= f_t.c_{t-1} + i_t.c_t^{'} \\
h_t &= o_t.\sigma_c(c_t)
\end{aligned}
$$

σ_g sigmoid, σ_c tahh,$< . >$ element wise multiplication.
f_t is the forget gate.
i_t is the input gate.
o_t is the output gate.
c_t is the cell stte.
h_t is the hidden state.
Where W_f, W_i, W_o, W_c andU_f, U_i, U_o, U_c are parameter matrices. The symbols x_t and h_t stand for the corresponding input embedding vector and hidden state vector at the time step t, respectively.

The encoder's task is to build the fixed length feature representation from the input sequence. In this regard, recurrent based bi-directional LSTM is used. It capture both the forward and

backward relationship between the words. The forward LSTM computes hidden state representations ($\overrightarrow{h_1}$, $\overrightarrow{h_2}$,... $\overrightarrow{h_j}$,... $\overrightarrow{h_m}$) at each word position sequentially in accordance with the current word embedding and the previous hidden state, given a sequence of the input word embeddings (i.e. (x1, x2, xj,... xm)]. For each word in reversed order, the backward LSTM generates hidden state representations ($\overleftarrow{h_1}$, $\overleftarrow{h_2}$,... $\overleftarrow{h_j}$,... $\overleftarrow{h_m}$) (i.e., from the last word to the first). The two different categories of hidden states are

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

Similary, cell states are defined as

$$\overrightarrow{c_t} = LSTM(x_t, \overrightarrow{c_{t-1}})$$
$$\overleftarrow{c_t} = LSTM(x_t, \overleftarrow{c_{t-1}})$$

Next, we initialize the Bi-LSTM's states to zero vectors, with $\overrightarrow{h_1} = 0$ and $\overleftarrow{h_m} = 0$ as well as $\overrightarrow{c_1} = 0$ and $\overleftarrow{c_m} = 0$. Each word in the input sequence can be represented as a concatenated hidden state of a forward LSTM and a backward LSTM using the notation h_t=[$\overrightarrow{h_t}$,$\overleftarrow{h_t}$]. Similarly, indicate as c_t=[$\overrightarrow{c_t}$,$\overleftarrow{c_t}$].
We use LSTM as the decoder to generate the output summary. The encoder and decoder constitute a basic sequence-to-sequence model. Then, we connect the encoder to the decoder, by passing these states [h_t,c_t].

We incorporate attention mechanism which takes the input from both encoder and decoder model as value and query respectively. In this regard, we give encoder output (h_t^e) and decoder output (h_t^d) to the attention layer. at=softmax($\frac{h_t^d . h_t^{e\top}}{\sqrt{d_{h_t^e}}}$).$h_t^e$
where at is attention output.

Then the output of the attention layer and decoder layer is passed to the concatenation layer. This concatenated output is then passed to time distribution layer, which calulates the probability distribution for each word in the summary dictionary.

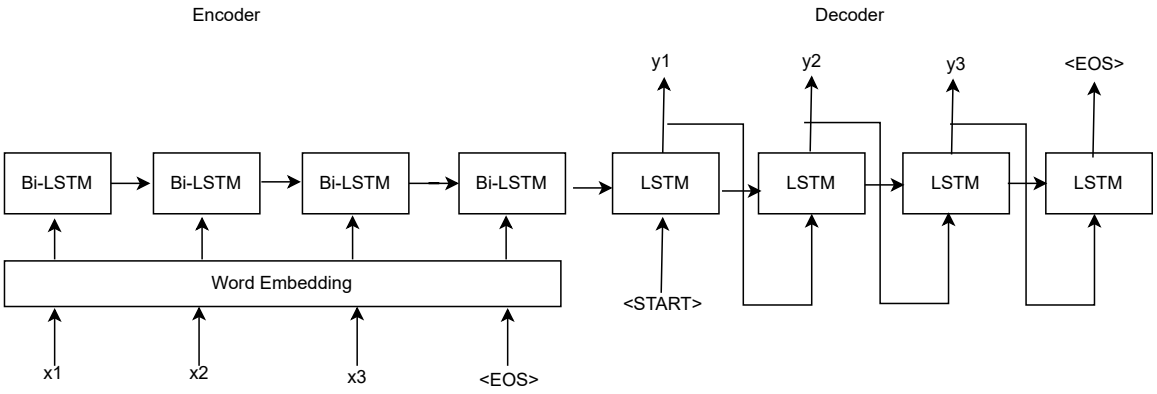
$P_v = P(y_i|y_1, y_{i-1}, x) = softmax(w_v.px_i + b_v)$
where $P(y_i|y_1, y_{i-1}, x)$ is the conditional probability distribution for the target word y_i over all words in the vocabulary at time step i, px_i is the input from the previous layer,and w_v and b_v are the learning parameters.

3.6. Sequence to Sequence Model

Every sequence-to-sequence model has an encoder that uses a bi-LSTM and a decoder that uses an LSTM architecture, as shown in Figure 3. The vocabulary now includes some fresh special tokens, such as "PAD," "EOS," and "START." Each PAD token helps to equalise the length of each sentence to a specific length. Each sequence contains the EOS token to signal the encoder model about the end of the statement. The start token instructs the decoder to begin

the decoding process. In this regard, we chose START and EOS from the data before our training data, which contains words that used sequence translation. Here, the encoder's input sequence is represented by x, and the generated output response or sequence is represented by y.

Figure 3: End-to-end sequence to sequence model



4. Experiment and Results

In this section, we discussed about the experimental setup, evaluation metric, and results.

4.1. Experimental setting

We have used tensorflow [15] version 2.9.1 for creating the proposed end-to-end sequence based model. Here, encoder Bi-LSTM has hidden state dimension = 128 and decoder lstm has hidden dimension = 256. The word embedding size = 300 is used. The proposed model will be able to generate a machine's own summary once we have finished training part. The proposed model generate the summary of length = 75. After evaluating and hyper-parameter tuning, we used rmsprop optimizer[16], sparse categorical crossentropy loss function for fast and better convergence, epoch = 50, batch size = 32, and learning rate = 0.001 as parameter for training the final model.

4.2. Evaluation Metric

We used the Recall-Oriented Understudy for Gisting Evaluation (ROGUE) [17] metric to evaluate our model. ROUGE counts the number of overlapping lexical units to evaluate the quality of generated summary. In this study, we use the scores like Rouge-1, Rouge-2, Rouge-3 and Rouge-4, which measure, how well the generated summaries match the actual summaries in terms of unigrams, bigrams, trigrams and 4-grams respectively.

4.3. Results

In this section, we discussed about the results obtained on validation and test datasets provided by ILSUM-2022 [3]. The produced results by proposed model on validation datasets {Hindi, English, and Gujarati} for Rouge-1 metric is shown in Table 2. Here, the proposed model used the embedding from the pre-trained model directly without updating those values during back-propagation (trainable=False) and overall lower parameters were trained in end-to-end network as shown in Figure 1. The proposed model produced results on test data {English} is shown in Table 3. The final results are obtained based on training the embedding during the back-propagation procedure (trainable = True). The evaluation metric such as precision, recall and F1-score has been evaluated for each Rouge-1 to Rouge-4. Rouge-1 is evaluated on 1-gram whereas Rouge-4 is evaluated on 4-gram , and so on. So, on increasing the n-gram from one to four, Rouge is decreasing for each performance metric. Our proposed model perform better for Rouge-1.

Table 2
Results on Hindi, English, and Gujarati Validation Dataset for Rouge-1 metric

Validation Datasets	Precision	Recall	F1-score
Hindi	0.294781	0.215508	0.238898
Gujarati	0.000111	0.000281	.000158
English	0.000616	0.000465	0.000961

Table 3
Results on English Dataset for Rouge-1, Rouge-2, Rouge-3 and Rouge-4

English TestData	Precision	Recall	F1-score
Rouge-1	0.356147961	0.326064401	0.328213556
Rouge-2	0.176917204	0.166501352	0.165805031
Rouge-3	0.129456619	0.123251567	0.122045746
Rouge-4	0.103489988	0.099644622	0.09809286

5. Conclusion and Future work

We extend the sequence-to-sequence framework for abstractive text summarizers by presenting a successful method for English-to-English text summarization using Bi-LSTM encoding and LSTM decoding layer. The foundation of our model is a straightforward encoder-decoder model with an attention mechanism. Extensive testing on the ILSUM-2022 (English dataset) demonstrates that our model generates state-of-the-art results.
In our upcoming work, we will intend to concentrate on how to balance precision and recall to further improve F1 performance by automatically choosing dynamic decoding length based on deep learning.

Acknowledgments

This first author would want to acknowledge the Ministry of Education (MOE), Government of India for financial support during the research work through the Rajiv Gandhi fellowship Ph.D scheme (UGC) for computer science & engineering.

References

[1] W. S. El-Kassas, C. R. Salama, A. A. Rafea, H. K. Mohamed, Automatic text summarization: A comprehensive survey, Expert Systems with Applications 165 (2021) 113679.

[2] S. Satapara, B. Modha, S. Modha, P. Mehta, Fire 2022 ilsum track: Indian language summarization, in: Proceedings of the 14th Forum for Information Retrieval Evaluation, ACM, 2022.

[3] S. Satapara, B. Modha, S. Modha, P. Mehta, Findings of the first shared task on indian language summarization (ilsum): Approaches, challenges and the path ahead, in: Working Notes of FIRE 2022 - Forum for Information Retrieval Evaluation, Kolkata, India, December 9-13, 2022, CEUR Workshop Proceedings, CEUR-WS.org, 2022.

[4] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1700–1709.

[5] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[6] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al., Abstractive text summarization using sequence-to-sequence rnns and beyond, arXiv preprint arXiv:1602.06023 (2016).

[7] H.-X. Pan, H. Liu, Y. Tang, A sequence-to-sequence text summarization model with topic based attention mechanism, in: International Conference on Web Information Systems and Applications, Springer, 2019, pp. 285–297.

[8] C.-Y. Lin, E. Hovy, The automated acquisition of topic signatures for text summarization, in: COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics, 2000.

[9] Z. Yang, Y. Yao, S. Tu, Exploiting sparse topics mining for temporal event summarization, in: 2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC), IEEE, 2020, pp. 322–331.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[11] H. Su, X. Shen, R. Zhang, F. Sun, P. Hu, C. Niu, J. Zhou, Improving multi-turn dialogue modelling with utterance rewriter, arXiv preprint arXiv:1906.07004 (2019).

[12] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, CoRR abs/1802.05365 (2018). URL: http://arxiv.org/abs/1802.05365. arXiv:1802.05365.

[13] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, Y. Bengio, Learning

phrase representations using RNN encoder-decoder for statistical machine translation, CoRR abs/1406.1078 (2014). URL: http://arxiv.org/abs/1406.1078. arXiv:1406.1078.

[14] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: https://www.tensorflow.org/, software available from tensorflow.org.

[16] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. Freitas, J. Sohl-Dickstein, Learned optimizers that scale and generalize, in: International Conference on Machine Learning, PMLR, 2017, pp. 3751–3760.

[17] F. Liu, Y. Liu, Exploring correlation between rouge and human evaluation on meeting summaries, IEEE Transactions on Audio, Speech, and Language Processing 18 (2009) 187–196.