Abstractive Text Summarization using LSTM

Apoorva Chavali, Clifford Reeve Menezes, Rutvik Dagadkhair, and Srinidhi Amuri

Virginia Tech, USA
ECE5424: Advanced Machine Learning

Abstract. The explosion of big data and technology has created a huge amount of information that it becomes difficult for individuals to process the information efficiently. It would be impracticable to have large texts, read and decide on relevance manually, since much of it might not be about the topic a reader is looking for. Regarding this, text summarization is one of the most important jobs in Natural Language Processing (NLP). It helps in summarizing gigantic chunks of information into short pieces while retaining much of the critical details. This project implements a deep learning-based abstractive text summarization model. We further used the sequence-to-sequence model architecture, an encoder which reads the source text and focuses on key information, and a decoder which then generates a coherent, condensed summary. A Long Short Term Memory (LSTM) network is utilized in both the encoder and decoder for capturing and retaining dependencies in the text over long ranges. By training the model and fine tuning it with a large and diverse dataset, we were able to generate a reasonably good summary for a given text.

Keywords: LSTM, Seq2Seq, Encoder, Decoder, RNN

1 Introduction

In the era of information overload, the ability to quickly distill key insights from large volumes of text has become increasingly valuable. Abstractive Text Summarization, a cutting-edge natural language processing (NLP) technique, aims to address this challenge by generating concise, coherent summaries that capture the essence of longer documents while potentially using novel phrasing. This project focuses on leveraging Long Short-Term Memory (LSTM) networks, a powerful type of recurrent neural network (RNN), to tackle the complex task of abstractive summarization. LSTMs are particularly well-suited for this task due to their ability to capture long-term dependencies in sequential data, making them ideal for understanding and generating human-like text. By employing LSTM architecture, this project aims to create a model capable of producing summaries that not only condense information but also demonstrate a level of language understanding and generation that goes beyond mere extraction of existing sentences. This research explores the intricacies of implementing LSTM-based abstractive summarization, from data preprocessing and model architecture design to training strategies and evaluation metrics. The project seeks to contribute to the ongoing advancements in NLP and machine learning, with potential applications ranging from content curation and news briefing to academic research assistance and business intelligence.

Out of a wide variety of sources for text that can include novels, blogs, websites, etc., this project will focus solely on news articles since it is a great source of rich grammar and diverse vocabulary. It is one of the application having high quality text growing everyday. Thus this study will consider news articles for model training and testing purposes.

2 Specific Aims

The main objective of the project is to develop a text summarization model with high accuracy and minimum loss. The particular objectives of the project are as follows:

- Development of a Sequence-to-Sequence Model for Text Summarization: Design an encoder-decoder architecture based on LSTM networks that will give concise and effective summaries from any input text.
- Optimization of Abstractive Summarization: Improve abstractive summarization such that the summarized content is paraphrased from the input text into its shortest form while considering readability and coherence.
- Improve Model Performance on Large-scale Data Sets: The implementation of the model is done on diverse large-scale text datasets for performance improvement in generating summaries that are correct and relevant within various domains.
- Fine tuning of the Model for Greater Precision: Perform hyper parameter tuning on the model for higher precision summaries which are informative with less loss.

3 Background

Text summarization is a task in Natural Language Processing that refers to the condensation of long texts into concise versions while retaining major information along with its meaning. It can be performed by two methods:

1. Extractive Summarization: It selects and extracts important sentences directly from the text.

2. Abstractive Summarization: It is a method of summarizing by rewording and condensing, much like humans create summaries.

With the volume of text published every day, summarization has become an important task in processing large volumes. It helps the reader read and comprehend a long document by providing a summary comprising the most useful information. For this reason, it turns out to be an indispensable tool in information-intensive domains.

4 Research Design and Method

4.1 Architecture

We used the Seq2Seq Encoder-Decoder[4] architecture based on LSTM[6]. It comprises two main components: the encoder and the decoder. The encoder processes an input text sequence and converts it into a fixed-length feature vector while also providing the final hidden and cell states of the LSTM. These states serve as the initial inputs for the decoder, which generates the summary sequentially.

The decoder uses an embedding layer to transform input tokens into vector representations and an LSTM layer to predict the next word in the sequence based on the current token, encoder output, and decoder states. A dense softmax layer generates a probability distribution over the vocabulary for the predicted word. The function returns two models: the encoder model, which encodes the input sequence, and the decoder model, which predicts the output sequence during inference.

The decoder LSTM performs inference for a single input sequence. It first encodes the input text to obtain the feature vector and initial states from the encoder. Starting with a sequence containing only the start token, the function iteratively predicts the next word, updates the decoder states, and appends the predicted word to the output sentence. The process continues until the end token is generated or the maximum summary length is reached.

Utility functions assist in converting numerical sequences back into readable text by mapping indices to words while excluding special tokens and padding. The predict text function orchestrates the summarization process for input text. It preprocesses and cleans the text, tokenizes it into numerical sequences, and generates the summary. For inputs exceeding the maximum allowable length, it splits the text into manageable chunks, summarizes each chunk individually, and combines the results into the final summary.

Overall, the implementation leverages the encoder-decoder framework to process input sequences and generate meaningful summaries in a modular and scalable manner, making it suitable for various text summarization tasks.

LSTM Layer Architecture- Recurrent Neural Networks use past experiences to improve neural networks performance on current and future inputs. The hidden state in an RNN stores past information and allows to operate on sequences. They have two sets of weights, one is for the hidden state vectors and the other one is for the current input. The output of an RNN architecture is based on both current input and hidden state vectors, that's based on previous input. LSTM networks are specialized form of the RNN architecture. LSTMs layers control the information that gets propagated from one hidden to another hidden state. This information is controlled by additional gates in the hidden states and these gates overcome the issue with RNNs in learning long-term dependencies. LSTM block has a memory cell, input gate, output gate and a forget gate. These gates allow the network to learn long-term relationships in the data more efficiently.

LSTM use two types of normalizing equations, i.e a sigmoid function and a tanh function. Sigmoid function is used to calculate set of scalars by which we multiply something else. Tanh function is used to transform data into a normalized encoding of the data.

The input, forget and output gates follow these general equations, where h_{t-1} is the copy of hidden state from previous time stamp, x_t is a copy of the data input at the current time-step, sigma represents sigmoid function w_x represents weight for the respective gate (x) neurons and b_x biases for the respective gates (x).

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$i_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{3}$$

Equations for input, forget and output gates in an LSTM are as described above
.

$$\tilde{c_t} = tanh(w_c[h_{t-1}, x_t] + b_c) \tag{4}$$

$$c_t = f_n * c_{t-1} + i_t * \tilde{c_t} \tag{5}$$

$$h_t = o_t * tanh(c^t) \tag{6}$$

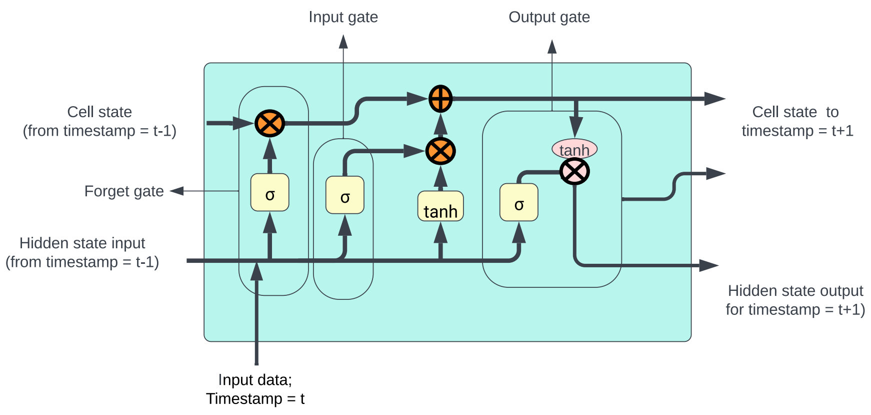

Fig. 1. LSTM architecture

The LSTM unit's behavior is regulated by different sets of weights and biases. At the input, these parameters determine how much new information is allowed to enter the unit. The forget gate's weights and biases influence how much of the existing information is retained within the unit. Finally, the output gate's parameters control which values are used to calculate the LSTM block's activation. Each of these gates plays a crucial role in managing the flow and processing of information through the LSTM unit.

The equation for candidate cell state, cell state and final output is given by:

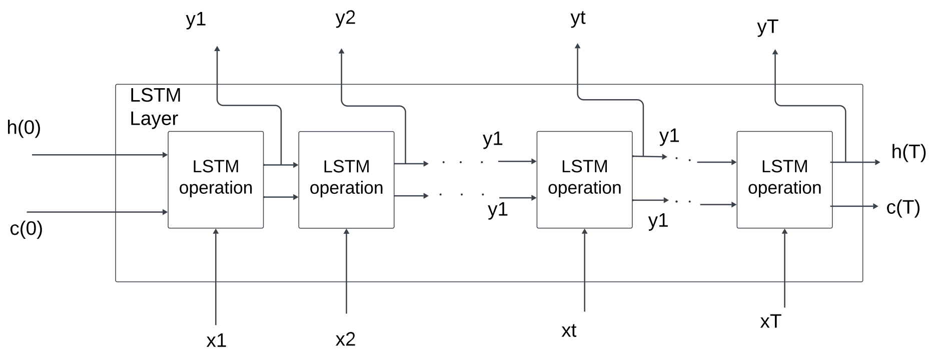

Fig. 2. Multiple layers of LSTM

Sequence to Sequence Learning (Seq2Seq model) Sequence to sequence models are an application of LSTM blocks that converts an input sequence of information to the output sequence. Sequence in this case can be symbols, words in a sentence. It consists of two processes:

1. From the input sequence X a fixed size vector z is generated.
2. The fixed-size vector z is processed to get a decode output sequence Y.

The information of [1] X is conveyed by z, and the probability $P_\theta(\mathbf{y}_j|\mathbf{Y}_{<j},\mathbf{X})$ can actually be computed as $P_\theta(\mathbf{y}_j|\mathbf{Y}_{<j},\mathbf{z})$. First, representing the process that generates z from X using the function Λ: $\mathbf{z} = \Lambda(\mathbf{X})$.

The functionΛ may be the recurrent neural net such as LSTMs.

Secondly, we represent the process which generates Y from z by using the formula:

$$P_\theta(\mathbf{y}_j|\mathbf{Y}_{<j},\mathbf{X}) = \Upsilon(\mathbf{h}_j^{(t)}, \mathbf{y}_j)$$
$$\mathbf{h}_j^{(t)} = \Psi(\mathbf{h}_{j-1}^{(t)}, \mathbf{y}_{j-1})$$

Here, Ψ is a function that generates the hidden vectors $\mathbf{h}_j^{(t)}$, and Υ is the function that calculates the generative probability of the one-hot vector $\mathbf{y}_j$. When $\mathbf{h}_{j-1}^{(t)}$ (or $\mathbf{h}_0^{(t)}$) is z generated by $\Lambda(\mathbf{X})$, and $\mathbf{y}_{j-1}$ (or $\mathbf{y}_0$) is the one-hot vector of BOS.

This code defines three callbacks in Keras to optimize the training process. The **EarlyStopping** callback monitors the validation loss (`val_loss`) and stops training if it doesn't improve for two consecutive epochs, helping to prevent overfitting and save time. The **ReduceLROnPlateau** callback adjusts the learning rate dynamically by reducing it by a factor of 0.1 if the validation loss stops improving for two epochs, ensuring that the learning process remains effective. Additionally, the **ModelCheckpoint** callback saves the model to a specified file (`best_model.keras`) whenever the validation loss improves, ensuring the best version of the model is preserved during training. Together, these callbacks enhance training efficiency, promote convergence, and safeguard the best-performing model.

Parameters used:

Hyper Parameters	Values
Learning rate	0.001
Dropout	0.4
Epochs	30

5 Dataset and Experiments

5.1 Data Collection

This project uses the 'news summary' dataset [3] that includes articles for various publishers. The dataset consists of 4515 examples and contains Author name, Headlines, Url of Article, Short text, Complete Article. The author gathered the summarized news from Inshorts and only scraped the news articles from Hindu, Indian times and Guardian. Time period ranges from febrauary to august 2017. For the project a standard split of 70 % for training ,10 % for validation and 20% for testing of the model.

Example: Here is a sample from the dataset:

– Author: Chhavi Tyagi
– Date: 03 Aug 2017,Thursday
– Headline: Hotel staff to get training to spot signs of sex trafficking
– Link: $http : //www.hindustantimes.com/india - news/rakshabandhan - compulsory - in - daman - and - diu - women - employees - to - tie - rakhis - to - male - colleagues$
– Text: The Administration of Union Territory Daman and Diu has revoked its order that made it compulsory for women to tie rakhis to their male colleagues on the occasion of Rakshabandhan on August 7. The administration was forced to withdraw the decision within 24 hours of issuing the circular after it received flak from employees and was slammed on social media.
– cText: The Daman and Diu administration on Wednesday withdrew a circular that asked women staff to tie rakhis on male colleagues after the order triggered a backlash from employees and was ripped apart on social media.The union territory?s administration was forced to retreat within 24 hours of issuing the circular that made it compulsory for its staff to celebrate

Rakshabandhan at workplace.?It has been decided to celebrate the festival of Rakshabandhan on August 7. In this connection, all offices/ departments shall remain open and celebrate the festival collectively at a suitable time wherein all the lady staff shall tie rakhis to their colleagues,? the order, issued on August 1 by Gurpreet Singh, deputy secretary (personnel), had said.To ensure that no one skipped office, an attendance report was to be sent to the government the next evening.The two notifications ? one mandating the celebration of Rakshabandhan (left) and the other withdrawing the mandate (right) ? were issued by the Daman and Diu administration a day apart. The circular was withdrawn through a one-line order issued late in the evening by the UT?s department of personnel and administrative reforms.?The circular is ridiculous. There are sensitivities involved. How can the government dictate who I should tie rakhi to? We should maintain the professionalism of a workplace? an official told Hindustan Times earlier in the day. She refused to be identified.The notice was issued on Daman and Diu administrator and former Gujarat home minister Praful Kodabhai Patel?s direction, sources said.Rakshabandhan, a celebration of the bond between brothers and sisters, is one of several Hindu festivities and rituals that are no longer confined of private, family affairs but have become tools to push politic al ideologies.In 2014, the year BJP stormed to power at the Centre, Rashtriya Swayamsevak Sangh (RSS) chief Mohan Bhagwat said the festival had ?national significance? and should be celebrated widely ?to protect Hindu culture and live by the values enshrined in it?. The RSS is the ideological parent of the ruling BJP.Last year, women ministers in the Modi government went to the border areas to celebrate the festival with soldiers. A year before, all cabinet ministers were asked to go to their constituencies for the festival[2].

5.2 Data Pre-processing

We parsed the dataset to remove unnecessary characters, converted all the characters to lowercase. Then we tokenized the text into tokens and padded the input sequence data length to a uniform size for short sequences or truncated the longer sequences. We used the pre-trained word embeddings like GloVe[5] and a randomly initialized embedding layer, to get the semantic relationships between words. Using these word embeddings we improved the model's understanding of the context.

5.3 Experiments :

These experiments shall help researchers and practitioners understand the capabilities, limitations, and possible improvements of abstractive summarization systems. This helps in advancing the research area and building summarization technologies that are truly effective and reliable.

Performance Evaluation	
Training accuracy	54.81%
Training loss	3.25011
Validation accuracy	55.25%
Validation loss	3.3080

- **Domain Adaptation:**

  We fine-tuned the model on domain-specific news data to evaluate the results.

- **Embedding Layer:**

  We experimented with a randomly initialized embedding layer as well as the pre-trained GLoVe embedding layer. We found that the randomly initialized embedding layer provides better results over the GloVe embedding layer. The randomly initialized embedding layer was trained specifically on the news summary dataset. This allowed it to learn task-specific word representations tailored to the dataset's context and vocabulary. GloVe embeddings, pre-trained on large corpora like Wikipedia or Common Crawl, represent general-purpose semantic relationships. These relationships might not align perfectly with the domain-specific context of the news summary dataset we used. GloVe embeddings could not adapt dynamically to the model's needs during training, had vocabulary coverage gaps and mismatched context.

- **Inference Time Experiments:**

  Length Control: When we experimented using multiple length values, we found that for higher values of length, the words were repeated and in some cases the output was gibberish. Whereas, for the lower values of length, the output was contextually unclear. After multiple trail and error scenarios, we found that a word length of 15 to give clear outputs.

6 Results and Discussions

The training and validation accuracy are as follows:

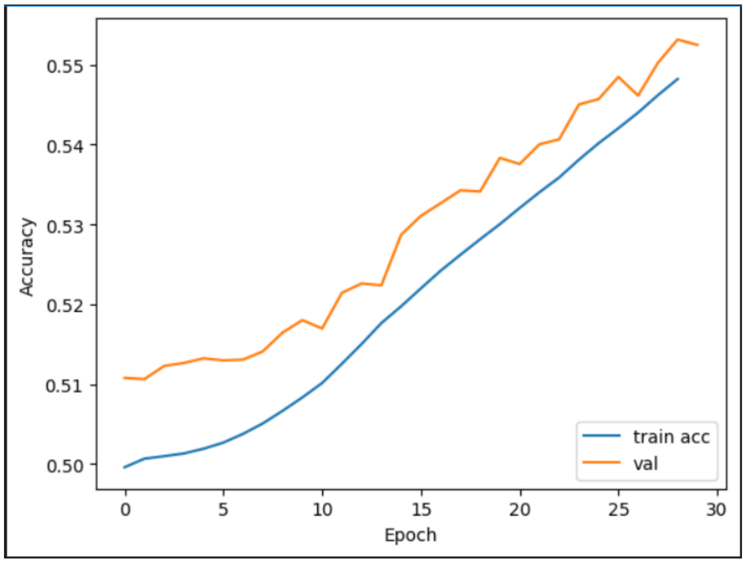

Fig. 3. Training and Validation Accuracy

The training and validation loss are as follows:

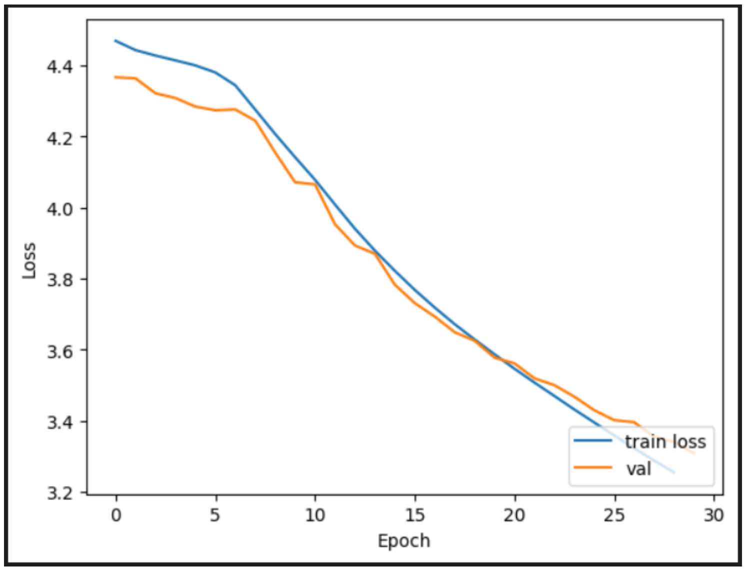

Fig. 4. Training and Validation Loss

The model predicted the following output for the given input:
News: train accidents last five years caused due derailments according government data derailment indorepatna express mber claimed lives people others injured recent derailment utkal express uttar pradesh killed left injured
Original summary: *start half train accidents last years due derailment end*
Predicted summary: *start railways get highest train due people end*

The accuracies suggest that while the model captures patterns, it is unable to learn complex relationships between input and output sequences. The losses indicate a possibility of underfitting. The randomly initialized embedding layer outerformed the pretrained GloVe embedding. This is because GloVe embeddings are general-purpose, capturing broad semantic relationships but lacking adaptability to domain-specific contexts.

For future iterations, we plan to experiment with Transformer-based architectures like BERT, T5 for improved contextual understanding. We plan to enhance the training by augmenting the dataset with more diverse samples or synthetic examples via back-translation. To adjust inferences, we plan to integrate convergence mechanisms to reduce word repetitions.

References

1. https://docs.chainer.org/en/v7.8.0/examples/seq2seq.html.
2. http://www.hindustantimes.com/india-news/rakshabandhan-compulsory-in-daman-and-diu-women-employees-to-tie-rakhis-to-male-colleague. 2017.
3. https://www.kaggle.com/code/koshirosato/text-summarization-with-simple-transformers-t5. 2020.
4. Caglar Gulcehre Dzmitry Bahdanau Fethi Bougares Holger Schwenk Yoshua Bengio Kyunghyun Cho, Bart van Merriënboer. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
5. Richard Socher Pennington, Jeffrey and Christopher D. Manning. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.
6. Sepp Hochreiter Jurgen Schmidhube. Long short-term memory. *Neural Computation*, 9:1735–1780, 2016.