The Scientific Temper (2023) Vol. 14 (3): 703-707

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2023.14.3.20

https://scientifictemper.com/

RESEARCH ARTICLE

A machine translation model for abstractive text summarization based on natural language processing

Bhuvaneshwarri Ilango

Abstract

"Knowledge is power and knowledge is liberating" conveys that there is a need for the capacity for creativity and that information is plentiful. The key application of natural language processing (NLP) is text summarization. It is a well-known technique for copying text, selecting accurate content, and get insight from the text. The purpose of this study is to propose for providing a summary of the text employing the seq2seq concept from the TensorFlow Python library. Through the use of deep learning-based data augmentation, the suggested method has the potential to increase the effectiveness of the text summary. Finally, the bilingual evaluation understudy (BLEU) criterion is used to judge the effectiveness of the suggested methodology.

Keywords: Machine translation model, Natural language processing, Summarization, Text.

Introduction

Numerous studies have been done in the field of abstractive text summarization, although they have always exclusively looked at neural network-based models. These strategies can now be used in conjunction with knowledge-based ones to make them more potent. This research proposes a sequence-to-sequence neural-based text summarising technique using the TensorFlow Python library (Gambhir, and Gupta, 2017). By addressing the problem of uncommon or out-of-vocabulary phrases, the suggested strategy can improve the usefulness of deep learning models. The idea behind is the generalization of content with deep learning-based summary of text. Many NLP tasks, such as speech recognition, machine translation, and producing captions for films, have been carried out using the Seq2seq paradigm. An encoder and a decoder are the two primary parts of the seq2seq paradigm. The primary responsibility of the encoder is to encode the context vector in order to save the data provided by the source text. According to the encoder's context vector, the decoder's task is to generate a target name for each time step. The core models, however, were rife with problems, such as wordiness, ambiguity, and the misuse of specific words in summaries. The attention mechanism generates an attention vector that aids the decoder by specifying which parts of the context vector should be given the most focus while producing a summary that keeps the context of the original article. The coercive teaching technique used by the teacher trains the decoder. It is required that come up with a similar word. The terms in the articles that need to be taught are thus replaced with words that have the same meaning with the aid of data additions. In this method, the words are changed, the total vector of the article is calculated using the modified sentence, and the decoder is compelled to produce words with related meanings. As a result, after the training procedure, the model contains grammatical sentences and may add new words to the sentences (Bhuvaneshwarri, 2020 and Bhuvaneshwarri, 2023).

Related work

NLP-based extractive text summarization was explored and studied by (Awasthi *et al.,* 2021. Based on linguistic and statistical criteria, they assessed the implications of statements.

(Chen *et al.,* 2018) described about text summaries using a semantic approach. A style transformation method that

Department of Information Technology, Government College of Engineering, Erode, Mettunasuvanpalayam, Tamil Nadu, India.

***Corresponding Author:** I. Bhuvaneshwarri, Department of Information Technology, Government College of Engineering, Erode, Mettunasuvanpalayam, Tamil Nadu, India, E-Mail: ibw@gcee.ac.in

How to cite this article: Ilango, B. (2023). A machine translation model for abstractive text summarization based on natural language processing. The Scientific Temper, **14**(3): 703-707.

Doi: 10.58414/SCIENTIFICTEMPER.2023.14.3.20

Source of support: Nil

Conflict of interest: None.

Received: 18/07/2023 **Accepted:** 20/08/2023 **Published** : 25/09/2023

Foaad Khasmod and colleagues proposed significantly impacted text transformation. After pre-processing, a tag is assigned for all words in the text. Following that, each line is divided into unique, non-overlapping sentence fragments. Wordnet will be used to check each fragment for possible replacements. Synsets are part of the lexical database Wordnet. One or more groupings of synonyms for a specific lemma make up the synsets. The synonyms of each term are so examined. Based on its hit rate, each synonym is ranked. The text is changed to reflect the term with the highest score.

(Mani and Maybury, 1999) discussed sophisticated methods for automatically summarising material. The current methodologies and applications in natural language processing were listed by (Montejo-Ráez and Jiménez-Zafra, 2022).

(Nallapati *et al.,* 2016) have been pointed out several models for abstractive summarization. The fundamental model consists of both an encoder and a decoder. It is constructed with the gated recurrent unit-recurrent neural network (RNN). The encoder and decoder work in two directions. This idea has primarily been modified by the vocabulary of decoder. It only applies to the source file text for the specified set. This reduces the size of the layer with soft-max of the decoder. The model is capable of capturing the primary concepts and entities with the use of extra directory search with detailed data sets. The language traits are accurately captured. A generator pointer approach is utilized to handle the lack of vocabulary terms. Suppose the specified word is present in the data set of training, the decoder finds out and then it is considered for further processing; suppose the specified word not present is noted in the original document and later used during the compilation of the summary. All of the aforementioned modifications are made to the core model in order to produce efficient summaries.

(Radev *et al.,* 2002) highlighted a number of text summarization-related concerns. For the purpose of abstractive sentence summarization, (Rush *et al.,* 2015) created a neural attention model. In their 2017 paper, (Verma and Lee, 2017) discussed extractive summarization based on heuristics. (Wazery *et al.,* 2022) talked about utilising deep learning to summarise Arabic material that is abstract. They deployed embedded models and measured their models using four assessment measures.

Deep learning key phrase generation-based seq2seq model was created by (Zhang and Xiao, 2018). They presented a design for seq2seq with key phrase generation. The encoder and decoder was fundamentally designed with unidirectional and bidirectional gated recurrent units. This design's primary goals are to address the summary's redundancy and lack of vocabulary words. The copy technique solves the out of vocabulary (OOV) problem. All words are divided into two pieces. The 25 most commonly used words and OOV vocabulary are included in the fixed vocabulary. In order to pick a word from the specified vocabulary or to copy the word directly from the source when producing a summary with soft switch method of decoder. Repetition is handled via the coverage mechanism. As a result of the potential for it to be perceived as a frame of memory, reducing repetition, and the words produced by a time step will be less similar.

Existing system and its drawbacks

The goal of existing systems and methods for abstractive text summarizing is to produce brief, coherent summaries that effectively convey the essential details of the source text [6-8]. They are as follows:

Transformer-based models

In terms of abstract text summarization, transformer models like generative pre-trained transformer (GPT) and bidirectional and auto-regressive transformers (BART) have produced encouraging results. These models have already been trained on sizable datasets and are optimized for particular summarizing tasks.

Pointer-Generator Networks

This method uses a hybrid model that can replicate words from the source text or create new words, combining extractive and abstractive methods. The model gains the ability to choose between creating a term from scratch or copying one from the input text.

Reinforcement learning

Some methods train the summarization model via reinforcement learning. The model is first developed using supervised learning, and it is then improved using reinforcement learning using incentive signals based on the ROUGE metric, which assesses the quality of summaries.

Pre-trained language models

The already trained language models similar to bidirectional encoder representations from transformers (BERT) and text-to-text transfer transformer (T5) can also be modified for a summary of tasks. These models' reliance on the input text to condition the generation allowed for competitive performance.

The limitations of the aforementioned models and the potential difficulties in providing overall coherence and flow in the summary are the main downsides of the current approach. It frequently includes material from the source text that is redundant and included in numerous sentences or paragraphs. Extractive models have trouble summarizing texts that aren't complete or coherent.

Proposed System

It is usual to train a model with sequence, which is frequently based on RNN or transformers, to provide abstractive text summaries that go beyond simple sentence extraction. As seen in Figure 1. An encoder plus a decoder make up this model. The encoder transforms the input source text into

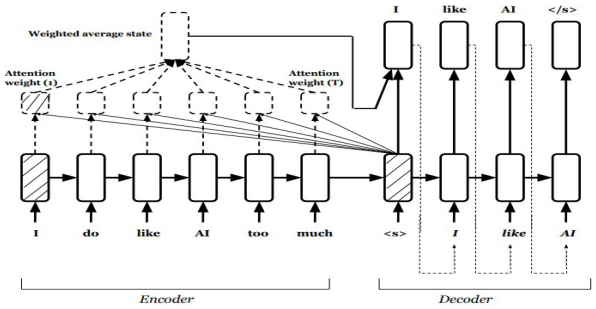

Figure 1: Seq2Seq with attention

a same and constant size representation while maintaining contextual information. Then, using the encoded form, decoder generates the summary. Throughout training, the model develops the capacity to map the original text to the desired summary. The review of the original text is provided with encoder, and using the encoder's output and the words that have already been produced, the decoder is taught to generate the summary word by word. In other models, like extractive summarization techniques, the summary is typically created by choosing and concatenating key phrases or sentences from the raw material. As a result, there may be a predisposition towards copying and repeating specific passages from the original text rather than coming up with fresh, abstract explanations. By creating summaries word by word, Seq2Seq models get over this problem and enable more inventive and unique summaries. Because Seq2Seq models can generate words on the fly, they can successfully handle OOV words, producing more thorough and precise summaries. Other models could concentrate on specific clauses or phrases, but Seq2Seq models consider the whole document. As a result, their summaries are more logical and contextually accurate because they are better able to identify significant linkages and dependencies between phrases.

Procedural steps

The procedural steps are depicted in Figure 2. Initial user input is usually a text file or other text that needs to be summarized. Pre-processing involves preparing the input text for subsequent analysis by removing stop words, punctuation, and other elements. Another stage in this process is arranging the text so the summarization algorithm can process it quickly. The central process, where the real summarization method functions, is abstractive summarization. It creates a succinct and cohesive summary of the given text using sophisticated natural language

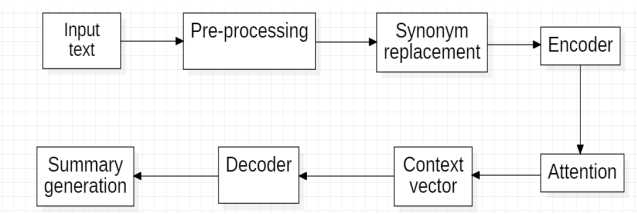

Figure 2: Procedural steps

processing techniques, such as transformer models or neural networks. The algorithm creates an abstract summary after analyzing the input text representation. Post-processing techniques are applied to the summarized output created output to improve readability and guarantee grammatical accuracy. These actions might entail trimming extraneous words, fixing grammatical problems, and enhancing the summary's overall coherence. The user is shown the abstracted summary of the input text, which is the final result.

System Implementation

Prerequisites

- Access to Google colaboratory requires a Google account.
- Basic knowledge of Python and deep learning concepts
- A dataset for text summarization (either download dataset from net or create own dataset)

Installation

- Create a new Python 3 notebook at https://colab.research.google.com.
- Install the required Python libraries. Run the following code in the first cell of the notebook:

  !pip install tensorflow==2.4.0; keras==2.4.3; numpy==1.19.3; pandas==1.1.5; nltk==3.5;
- Download the NLTK library
- Import the required libraries. Run the following code:

  import numpy as np
  import pandas as pd
  import re
  from bs4 import BeautifulSoup
  from keras. pre-processing.text import Tokenizer
  from keras.utils import pad_sequences
  from nltk.corpus import stopwords
  from tensorflow.keras.layers import Input, LSTM, attention, embedding, dense, concatenate, time distributed from tensorflow.keras.models import model
  from tensorflow.keras.callbacks import Early Stopping
  import warnings
- Load and pre-process the dataset. This may vary depending on the format and structure of dataset
- Utilise the Seq2Seq with Attention to build the model architecture.
- Train the above using the pre-processed dataset.

Module Implementation

The Seq2seq paradigm changes one sequence into another sequence. In order to circumvent the gradient vanishing circumstance, recurrent neural networks (RNNs), or more typically LSTM or gated recurrent unit (GRU), are utilised in this process. Each item's context is provided by the output from the stage before it. In order for RNNs to find abstractive

and extractive text summaries and compare them with various high-level methodologies, this study largely focuses on adding context as a first step. We employ labeled and sparsely labeled data to extract it. We train models using titles and subtitles to quote. Simple RNNs, LSTMs, or GRUs can be used as the RNNs in the encoder and decoder. Every hidden state in a straightforward RNN is calculated using the formula (1) as given below,

Ht(encoder)= ϕ(WHH*Ht-1+WHX*Xt) (1)

where is the hidden states connection matrix of weights, is the input and the hidden states connection matrix of weights, ϕ, the activation function, and represents the hidden states in an encoder. The following can be used to calculate the decoder's hidden states in formula (2),

Ht(decoder)= ϕ(WHH*Ht-1) (2)

The first state of the hidden of decoder is the final state of hidden retrieved from the encoder. The decoder's output is presented as follows in formula (3),

Yt= Ht(decoder)*WHY (3)

where WHY is the hidden states connection matrix of weight with the decoder output.

Working procedure for the implementation of the proposed system

The following stages are involved in the deployment of a system for abstractive summary of text using a seq2seq model:

Data Preparation

Summaries of paired source texts and references have been compiled into a dataset. The data has been cleaned, tokenized, and fragmented into train, validation, and test data sets as in pre-processing phase. Vocabulary mappings have been produced that provide each word or token in the dataset a distinct numerical index.

Model Architecture

A seq2seq model architecture, such as an LSTM-based or transformer-based model has been chosen. The model's hidden units, layers, and other hyperparameters have all been determined. The encoder and decoder components using a deep learning framework like TensorFlow or PyTorch have been implemented.

Embedding and Input Encoding

The text has been pre-processed by tokenizing it into words, sub-words, or characters. The input text using the encoder component, which can consist of multiple LSTM or Transformer layers has been encoded.

Decoding and Summary Generation

The decoding procedure has been put in place to form the summary of words. The decoder has been initialized with the encoded representation and a started hidden state.

Training

The training pipeline using the pre-processed dataset has been set. A loss function, such as cross-entropy sequence-to-sequence loss, has been defined to quantify the difference between the summaries of created and reference pair.

Evaluation

The summary comparison between created and reference has been evaluated using evaluation criteria such the BLEU score. Analyze the model's performance on the validation set on a regular basis to track development and adjust hyperparameters as necessary.

Fine-tuning and Optimization

The model using the training dataset specific to the target summarization task to improve its performance has been fine-tuned.

Results and Discussion

The seq2seq model of TensorFlow Python's library is used to make abstractive text summarization. The effectiveness of the suggested model is increased by augmenting the data with deep learning-based text summaries. Figure 3 displays the developed and correctly trained model. Figure 4 displays the suggested model's whole execution sequence.

The outcome is shown in Figure 5. Finally, the system quality is assessed with the Bilingual Evaluation Understudy

Figure 3: Training

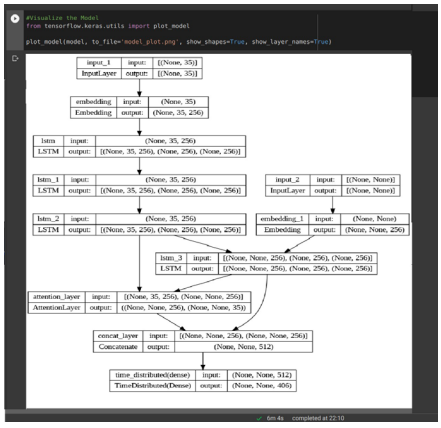

Figure 4: Visualization

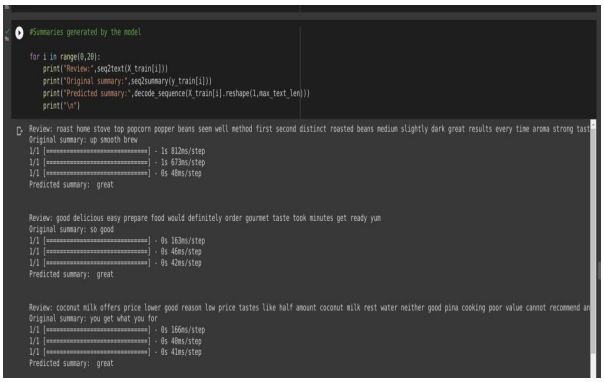

Figure 5: Result

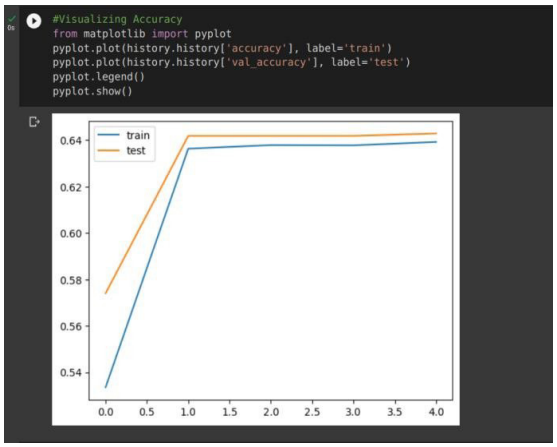

Figure 6: Accuracy graph

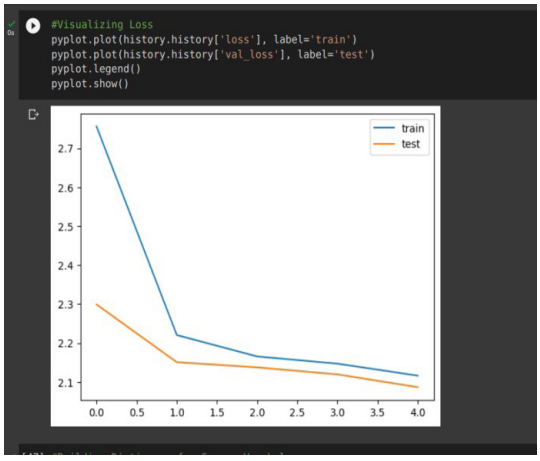

Figure 7: Loss graph

(BLEU) standard. This is depicted in Figures 6 and 7. These figures unequivocally demonstrate that the test model's accuracy is higher than train model and that its loss rate is lower. It implies that the suggested system achieves effective abstractive text summarization.

Conclusions

A promising method to create summaries that go beyond simple sentence extraction is abstractive text summarization using Seq2Seq models. Seq2Seq models, made up of an encoder and a decoder, develop the ability to map the source text to the intended summary by gathering contextual data and producing word-by-word summaries. These models can successfully manage long-range relationships, choose crucial information, and produce coherent and succinct summaries by utilizing techniques like word embedding, attention mechanisms, and beam search. In natural language processing, producing a summary of the abstractive text that captures the essence of the input content can be challenging. There are many fields in which abstractive text summarization technology could be used. Therefore, the future potential is bright. By decreasing the time consumption, summarization can be enhanced further.

References

Awasthi, I., Gupta, K., Bhogal, P.S., Anand, S.S., Soni, P.K. (2021). Natural language processing (NLP) based text summarization-a survey. 6th International Conference on Inventive Computation Technologies (ICICT) 2021, IEEE: 1310-1317.

Bhuvaneshwarri, I. (2020), Android Geo-Location Based Smart Bus Ticket Booking System, Iconic Research and Engineering Journals,7(1):166-169.

Bhuvaneshwarri, I. (2023), Determination of factors affecting stock market analysis during war, pandemic period using rough set and scalable future stock market price prediction model, Gradiva Review Journal, 9(6): 1137-1143.

Chen, P., Wu, F., Wang, T., Ding, W. (2018) A semantic QA-based approach for text summarization evaluation. In Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

Gambhir, M, Gupta, V. (2017). Recent automatic text summarization techniques: a survey, Artificial Intelligence Review, 47(1):1-66.

Mani, I., Maybury, M.T. (1999), Advances in automatic text summarization. MIT Press.

Montejo-Ráez, A., Jiménez-Zafra, S.M. (2022). Current Approaches and Applications in Natural Language Processing, Applied Sciences, 12(10):4859.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang,B. (2016). Abstractive text summarization using sequence- to- sequence rnns and beyond. Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, August 2016, Berlin, Germany, p. 280-290.

Radev, D.R., Hovy, E., McKeown, K. (2002). Introduction to the special issue on summarization, Computational linguistics, 28(4): 399-408.

Rush, A.M., Chopra, S., Weston, J. (2015). A neural attention model for abstractive sentence summarization. arXiv preprint arXiv: 1509.00685.

Verma, R., Lee, D. (2017). Extractive summarization: Limits, compression, generalized model and heuristics. Computacion y Sistemas, 21(4): 787- 798.

Wazery, Y.M., Saleh, M.E., Alharbi, A., Ali, A.A. (2022). Abstractive Arabic text summarization based on deep learning. Computational Intelligence and Neuroscience.

Zhang, Y., Xiao, W. (2018). Keyphrase generation based on deep seq2seq model. IEEE Access, (6): 46047-46057.