IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Abstractive Text Summarization Using Attention

[1] A.V. Patil, [2] Prachi Pophale, [3]Pooja Fattepure, [4]Umera Attar, [5]Vikas Raskar

[1]Professor, [2]Under Graduate, [3]Under Graduate, [4]Under Graduate, [5]Under Graduate

[1]Information Technology,

[1]Zeal College of Engineering and Research, Pune, India

Abstract:

The world we live in is full of text which is increasing day by day but people these days don't have time to read big paragraphs, articles, reviews or any larger body of text. People want to grasp things but in small summaries. So, our project aims to solve this problem by providing short summaries of the given text using Natural Language Processing that is NLP. Among the many tasks of NLP, text summarization is the most difficult task especially Abstractive summarization. Many Extractive summarization models based on text rank algorithm are available but their summaries are just collection of important words from the text and are not humane, like those of Abstractive model. In this project we have implemented Attention-based Seq2Seq (LSTM/GRU) and Transformer based Abstractive Summarization models which would summarize paragraphs, articles, reviews or any English text in abstractive manner. As a part of UI for the user, a webpage is shown wherein a provision to enter the text is provided. The text is then used as an input to our respective models and the result is an abstractive summary corresponding to the user input.

I. INTRODUCTION

Summary is considered as the important sentences from the original text. In an abstractive text summarization [1], system generates new phrases by interchanging phrases or using words that are not in the original text. In past few years we are witnessing the textual information is growing rapidly. It became more tough for the user to read the textual information and also it leads to loss of interest. Because of this text summarization came into picture which will solve this problem.

Text Summarization is process of generating a short text from long textual documents while maintaining important facts. Focus is to create well organized and fluent summary which have only important points outlined in the document. Most of the Machine Learning models trained to understand textual documents and filter the useful information before outputting the required summarized texts. Automatic text summarization is well known problem in machine learning and natural language processing [2]. Text Summarization has two approaches:

1) Extractive text summarization
2) Abstractive text summarization

In extractive text Summarization, system generates summaries by copying sentences from source text document through some measure of importance and then combine those sentences together. Importance of sentence is decided through linguistic and statistical features.

In abstractive [3] text summarization systems generate new phrases by interchanging phrases or using words that are not in the original text. Abstractive approach is harder than extractive text summarization. To generate effective abstractive summary the model has to first understand the text document then try to convey that understanding in a short possible way using new words and phrases.

II. OBJECTIVE

The purpose of this project is to provide short summaries of the given text using NLP based Deep Learning Models. This model would be beneficial for any person like teacher, student, scientist, etc. who doesn't have time to read large body of text and want the gist of it. This project could also be helpful for developing summary engines/websites where the user could get summaries of any type of text. This project could act as a base for future improvements in the field of Natural Language Processing, Sequence-to-Sequence models and Transformer models.

III. SCOPE

This project is developed to help the users to generate short summary of long text. This project can be used for educational purpose also to understand the context of any text without spending much time to read everything. With the help of this project, users can understand context and save the time.

IV. LITERATURE SURVEY

Many natural language processing related tasks such as text summarization [4, 5], Image captioning and Machine translation [6, 7, 8] are successfully implemented using Seq2Seq models. At first, Rush et al [9] used attention-based encoder and neural network language model (NNLM) decoder and introduced neural attention Seq2Seq model. In addition to this model, Chopra et al [10] further enhanced this model by replacing the feed forward NNLM with RNN. Some new concepts were introduced by Nallapati et al [5] to the RNN encoder decoder architecture to address different problems related to abstractive text summarization. It included 1) A special encoder to capture keywords 2) A switching generator-pointer to detect and handle Out of vocabulary words. Also, many models tried to summarize short documents into single line summary [9, 10]. These models have several shortcomings when it comes to summarize long documents into multi-sentence summaries: 1) They cannot accurately reproduce the salient information of source documents. 2) Unable to handle OOV words efficiently. 3) Generally, suffers from generating non repetitive summary and end up looking unnatural.

To solve the first two challenges, a pointer-generator network method was proposed by see et al [4]. It can copy a word from source text via pointer and then find a new word for it using generator. With the help of this technique, actual information can be reproduced and also OOV words are handled effectively. To address the third problem, various mechanisms are used such as coverage mechanism [4], intra-temporal and intra-decoder attention mechanisms [11].

V. METHODOLOGY

Most of the neural sequence models are based on encoder-decoder architecture. Encoder is responsible for generating a continuous representation of input text. Decoder basically takes the output of encoder and generates output sequence of it. This process is carried out by one element at a time. In this process, the model is self-enhancing and consuming the previously generated results and getting better. The Transformer is a transduction model which is entirely based on an attention mechanism to draw global dependencies between input and output. It replaces recurrent layers mostly used in encoder-decoder architectures with multi-headed self-attention layers. It allows for significantly more parallelization so it can be trained significantly faster than recurrent or convolution-based architectures. Figure 1: Source [2]

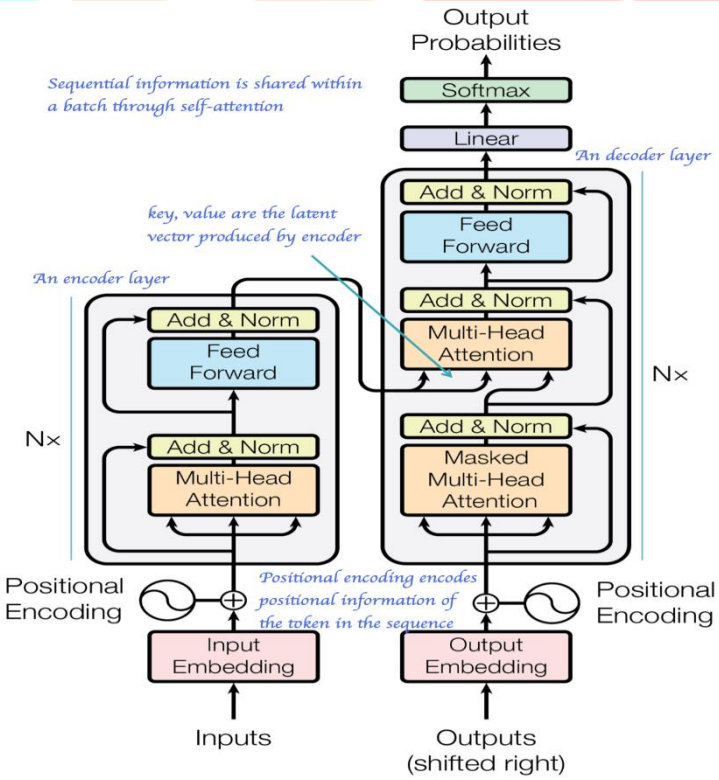


Figure 1: Model Architecture

Encoder: The encoder consists of a stack of N = 6 identical layers. every layer has 2 sub-layers. the primary is a multi-head self-attention mechanism, and the second is a position wise connected feed-forward network. we tend to use a residual affiliation around each of the 2 sub-layers, followed by layer normalization. That is, the output of every sub-layer is LayerNorm(x + Sublayer(x)), wherever Sublayer(x) is that the performed by the sub-layer itself. To facilitate these residual connections, all sub-layers within the model and the embedding layers, manufacture outputs of fixed dimension.

Decoder: Just like the encoder, decoder too is comprised of a stack of N = 6 identical layers. Apart from the 2 sub-layers in every encoder layer, the decoder inserts a 3rd sub-layer, that performs multi-head attention over the output of the encoder stack. just like the encoder, we tend to use residual connections around every sub-layer, followed by layer standardization. we tend to additionally modify the self-attention sub-layer within the decoder stack to stop positions from progressing to later positions. This masking, combined with incontrovertible fact that the output embeddings square measure offset by one position, ensures that the predictions for position i will rely solely on the notable outputs at positions less than i.

Self-Attention Model: Attention simply maps a query and a group of key-value pairs to an output that is calculated as a weighted addition of the values, wherever the compatibility function of the query with corresponding key is used to calculate the weight assigned to each value. Query, keys, values and output, these all are used as vectors. There are two kinds of attention functions. First is additive attention. It computes the compatibility operation employing a feed-forward network with one hidden layer. Second is Dot-Product attention. It computes by applying SoftMax on the dot products of the query of dimension d_k with all keys of dimension of d_k and so multiplying those SoftMax scores with values of dimension d_v.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where Q is matrix of set of query vectors, K is matrix of set of key vectors and V is matrix of set of value vectors.

$$Softmax\ S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

In this paper, Scaled Dot-Product Attention is used where the dot-products of query with all keys is divided by the scaling factor $\sqrt{(d_k)}$ and then SoftMax function is applied to obtain weights on the values. In comparison to additive attention, dot-product attention is faster and more space-efficient due to optimized matrix multiplication code. Figure 2: Source [1].

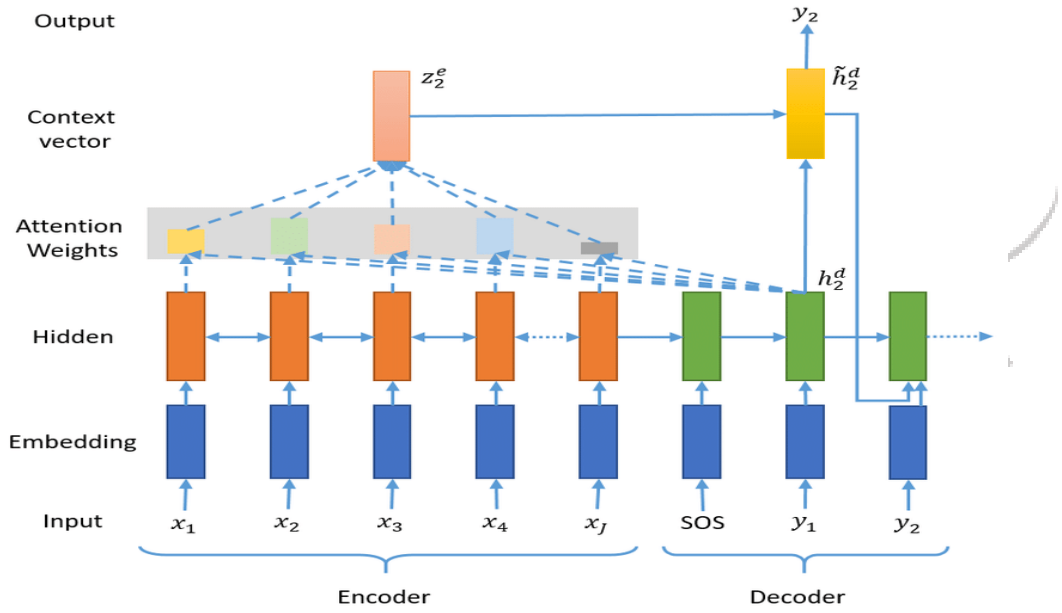


Figure 2: Sequence-to-Sequence model with Attention

Multi-Head Attention: In multi-head attention, the queries, keys and values square measure linearly planted h times with totally different learned linear projections to d_k, d_k, and d_v dimensions. Then Attention function is performed parallelly on each of these projected versions of queries, keys and values giving output values of dimension d_v. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions while with a single attention head this is not possible.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameters matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$

POSITION-WISE FEED FORWARD NETWORK

Apart from the attention layers, the encoder and decoder also contain the feed forward network. This network is applied to each position separately and in an identical manner. Two linear transformations with a ReLU activation in between is used.

$$FFN(x) = \max (0; xW1 + b1) W2 + b2$$

Different parameters are used from in each layer even though the linear transformations are the same throughout the different positions. It can also be described as two convolutions with kernel size 1.

Adam Optimizer

Adam (adaptive moment estimation) is an optimisation algorithm which optimises first-order gradient-based stochastic objective functions, which are based on adaptive estimates of lower-order moments. Implementing this optimiser is straightforward, computation wise efficient, memory requirements are less, is invariant to diagonal while rescaling of the gradients, and is best suited for large data problems. Using this optimiser is also suitable for non-stationary objectives and solving problems related to very noisy and/or sparse gradients. The hyper-parameters typically require little tuning and intuitive interpretations.

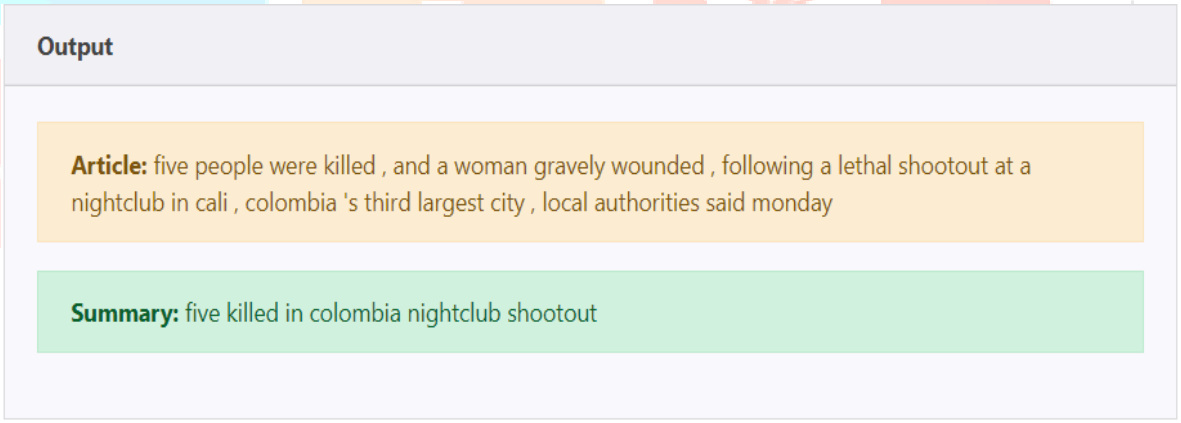
Adam is designed after considering the advantages of two popular methods and combination of these two: AdaGrad, which works best for sparse gradients, and RMSProp, which works best in on-line and non-stationary settings. Adam has several advantages like the while gradient rescaling magnitudes of parameter updates are invariant. Adam optimiser's step sizes are approximately bounded using step size hyperparameter, works on sparse gradients, and performs a form of step size annealing.

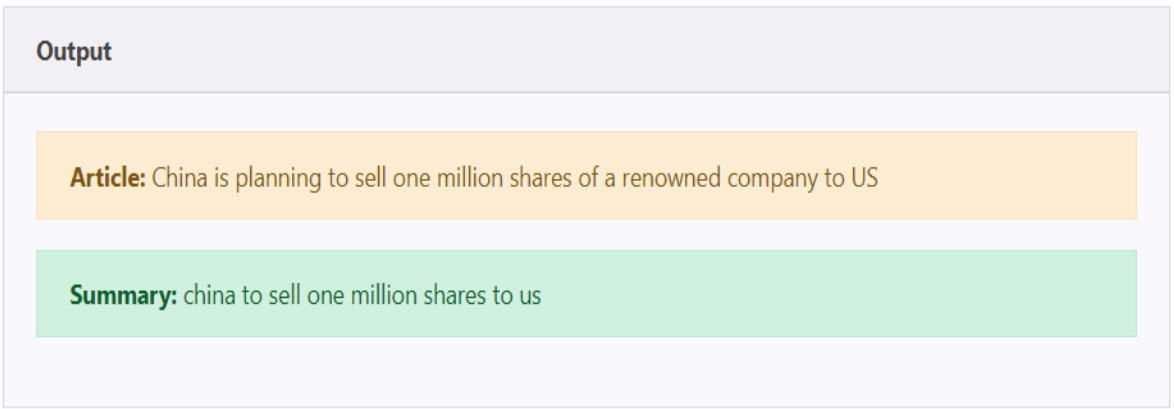
Required Parameters with optimal values:

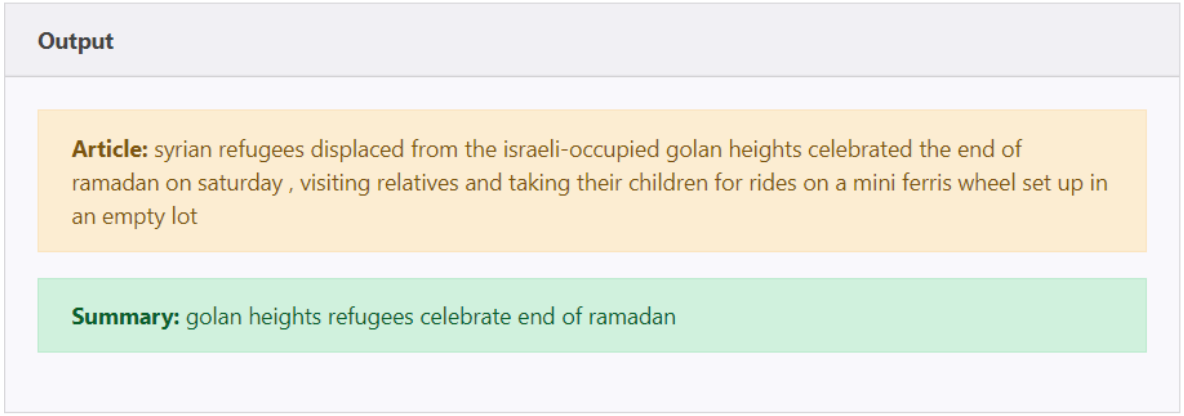
- α — It denotes the step size parameter (10^{-3}).
- ε — this parameter prevents *Division from zero* error (10^{-8}).
- β1 — Useful in decaying the running average of the gradient (0.9).

β2 — Useful in the running average of the square of gradient

Results







References:

[1] Shi, Tian, et al. "Neural abstractive text summarization with sequence-to-sequence models." ACM Transactions on Data Science 2.1 (2021): 1-37.

[2] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[3] Rush, Alexander M., et al. "A neural attention model for sentence summarization." ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing. 2017

[4] A. See, P. J. Liu, and C. D. Manning, “Get to the point: Summarization with pointer-generator networks,” in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, 2017, pp. 1073–1083.

[5] R. Nallapati, B. Zhou, C. dos Santos, C¸. glar Gulc¸ehre, and B. Xiang, “Abstractive text summarization using sequence-to-sequence RNNs and beyond,” CoNLL 2016, p. 280, 2016.

[6] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421.

[7] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” arXiv preprint arXiv:1609.08144, 2016.

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

[9] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 379–389.

[10] S. Chopra, M. Auli, and A. M. Rush, “Abstractive sentence summarization with attentive recurrent neural networks,” in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 93–98.

[11] R. Paulus, C. Xiong, and R. Socher, “A deep reinforced model for abstractive summarization,” arXiv preprint arXiv:1705.04304, 2017.