Introduction to Deep Learning and its Applications in Text Summarization

Raksha Ramesh
Department of Computer Science
PES University
Bangalore, India
raksharuby@gmail.com

Swanuja Maslekar
Department of Computer Science
PES University
Bangalore, India
swanuja2000@gmail.com

***Abstract*—As the amount of information available to us has been increasing exponentially, text summarization has increasingly become more important as it reduces the time required to comprehend information. In this paper we aim to describe and explain the basics required to understand automatic text summarization which includes neural networks and how it forms the basis of deep learning, followed by a method to achieve the same.**

***Keywords*—text summarization, natural language processing, extractive summary, abstractive summary, neural networks, deep learning**

I. Introduction

Text summarization is the process by which a large pool of sentences, like an article or report is analysed and a comprehensive summary is returned. It is used and applied in numerous fields, for example, it is used by news agencies to generate headlines from their articles, it is used in question-answer bots to condense the found information into a small summary that is presented to the end user and it is used to summarize books, to analyse research papers and other forms of literature, and so on. Text summarization can be broadly divided into two categories - extractive summarization and abstractive summarization, where the former is achieved by sentence extraction and the latter is achieved by statistical analysis.

Text summarization falls under the umbrella of Natural Language Processing (NLP). NLP is broadly defined as the automatic manipulation of natural language, like speech and text, by software and its implementation is being done by statistical methods which make use of machine learning, thereby delivering higher quality summaries with great accuracy and speed. In more recent times, NLP is being implemented, especially in abstractive summarization, by using deep learning for which understanding artificial neural networks and their types is extremely important.

II. Difference between Machine Learning and Deep Learning

In a machine learning (ML) model, there will be an algorithm that analyses and understands the data and learns from it and then applies whatever it has learnt from the given data on a new set of data to give an output, which could be a prediction or a conclusion. A machine learning model requires structured data to work with. This model becomes better over time as it studies the given data more and more. If the output given by this model is incorrect, it requires a human to make changes in the algorithm to correct the error as a machine learning model doesn't correct or learn from its mistakes by itself. Machine learning has numerous applications in a vast number of fields. A useful application of it is that it can be used as a recommending application. A user's activities on that specific application can be supplied as data to the ML model and based on what the user is interested in, it can recommend similar and newer things to the user. The model would have associated the given data with other users' data who have a similar field of interest.

Deep learning (DL) is a more advanced and evolved form of machine learning, it functions in a manner similar to machine learning. In a deep learning model, there will be numerous layers of an algorithm, each layer working on the data provided to it and giving an output with different interpretations, for the next layers to work with. This structure of having layers of an algorithm is called an 'artificial neural network'. This neural network is what makes deep learning different from machine learning. This model decides for itself whether its output is correct and fixes itself if it's incorrect. It does not require a human to physically make changes in the algorithm to give the correct output.

The difference between machine learning and deep learning can be further explained by a problem statement where the model needs to identify whether the given image is that of a specific animal, for example - an elephant, or not. In an ML model, organised and specific data which identifies the distinct features of an elephant need to be fed to the model manually. The model will then identify these features in the given image and give an output. In a DL model, there is no need to provide such structured data and can identify the animal from the image by passing it through the layers of the neural network.

The main differences between ML and DL can be summed up in the following sentence - ML requires organized data and human interference to correct its errors whereas DL does not as it can make intelligent decisions by itself and learns and corrects itself due to the presence of neural networks.

III. Extractive & Abstractive Summarization

There are mainly two methods of automatic text summarization, extractive and abstractive. Extractive summarization is achieved by picking the top sentences from the text based on some ranking system and presenting these sentences as the summary. Abstractive on the other hand involves understanding and comprehending the source text in a linguistic manner and generating a completely new set of sentences to effectively represent the summary of the source text.

There are different extractive methods, some better than the others, however, it is seen that an abstractive approach gives a summary with better focus and keeps the redundancy to a minimum while maintaining a good compression rate.

IV. Extractive Summarization

In extractive summarization, the important sentences from the document are recognised and extracted. All these selected sentences are then rearranged and combined to form a new and summarized document. No new sentences are generated by the summarization model; the summary simply consists of the original sentences which were identified to be important.

There are three ways to conduct an extractive summarization - frequency based approach, feature based approach and machine learning based approach. These three approaches are explained briefly below -

A. Frequency based approach

In this approach, it is assumed that the more number of times a word appears, the more important it is, that is, words of higher significance will appear more number of times than the insignificant ones. Hence, we use 'word frequency' to find and extract the important sentences in a document. This can be done in two ways -

1) Word probability: In this technique, the number of times a specific word appears in a document is simply counted and compared against the total length of the document. This is called the 'probability of a word' and is the ratio of the frequency count of that word in the document to the total number of words in the document. Using this ratio, it then computes 'sentence weights' for each sentence and then picks the top sentences based on the 'sentence weight' values.

2) Term Frequency-Inverse Document Frequency: This technique reduces the weightage of term frequency (tf) which is equal to word probability by dividing it with the inverse document frequency (idf). This value is calculated by dividing the total number of documents by the total number of documents that contain that specific word. The tf-idf ratio is then what is used to calculate weights and to decide which sentences are important.

B. Feature based approach

In this approach, features such as sentence position and presence of title words are used to identify the important sentences. Some of the important features are title words - if the words present in the title appear in a sentence, that sentence is most likely to be important, sentence position - beginning sentences and ending/concluding sentences are most likely to be important, sentence length - short sentences may be less significant compared to the longer ones, key words and so on. These features have 'scores' and 'weights' computed and the sentence score is then calculated using this which helps to extract the top sentences.

C. Machine learning based approach

The working of this approach is similar to any other machine learning model. There will be a training dataset which will consist of sentences marked as 'summary' or 'non-summary' sentences. The model will learn from this training dataset and then can identify the potential summary sentences in the new dataset that will be passed through it.

V. Artificial Neural Network

An artificial neural network (ANN) is the backbone of a deep learning model. It is seen as a 'black-box', where some data goes into this network and we then get the required output at the other end of the network; without knowing what exactly is happening in between. A neural network can be defined as - "a computing system that consists of a number of simple but highly interconnected elements or nodes, called 'neurons', which are organized in layers which process information using dynamic state responses to external inputs". Explained more simply, a neural network is designed and works in a way similar to the human brain. It consists of neurons, the same way our brain does, which are processing units and are highly interconnected to other neurons, again, similar to our brain.

A neural network consists of an input layer, hidden layers and an output layer. All the layers in between the input and output layer which do not take input or produce output are simply called the hidden layers. All the data processing happens in these hidden layers. A node is connected to several nodes in the layer below from where it receives data and is connected to several nodes in the layer above where it sends data to and hence is called 'feed-forward', that is, data moves through the network in only one direction.

A brief explanation of how neural networks work - The input layer has one neuron for each component of the input data. This input data is introduced to the input layer and then passed onto the hidden layers. Each of the connections have a 'weight' and 'bias' assigned to it. These weights and biases are initially set to random values which get adjusted continuously as and when the network is being trained by the training data. A neuron receives an input, it calculates the weighted sum which includes the bias and then uses a pre-set activation function (mostly sigmoid, more recently the ReLu). This finally results in a single number. If the number is below the threshold value, no data is passed to the next layer. If the value is above the threshold value, the neuron 'fires' its connections. The neuron then transmits all the data to its connecting neurons in a process called the 'forwards-pass'. At the end of this process, the last hidden layer is connected to the output layer which also consists of neurons, one neuron for each of the possible outputs.

VI. An Example of an Artificial Neural Network – Recognising Hand-Written Digits

The concept can be more thoroughly understood with a simple example. Let the neural network in example be an ANN to recognise hand-written digits.

Let the input data be a 20X20 pixel image of the hand-written digit. The neural network will start with a bunch of neurons, 400 to be precise; each neuron corresponding to each of the 400 pixels. These 400 neurons make up the first layer of the network. The last layer in this network will have 10 neurons, each neuron corresponding to each of the 10 digits. All the layers in between these two layers are the hidden layers. Let there be two hidden layers with 16 neurons in each for this example.

The neurons will hold a number whose value will be that of the grey scale, that is, between 0 (black) and 1 (white). This number is also called its 'activation number'. High activation means white, which means the neuron is lit up. Activation in one layer will decide the activation in the next layer. The activation of the neurons in the last layer represents how much the given system thinks corresponds to a given digit.

If an image is fed to the network, lighting up all 400 neurons of the input layer according to the brightness of each pixel in that image, this pattern of activation will trigger a very specific pattern in the next layer and this

continues to happen till it reaches the output layer and gives an output, where the brightest neuron in the output layer represents the digit.

Ideally, the layers would try to recognise the subcomponents of the hand-written digit and would then group them together to recognise the digit. For example, if the hand-written digit is '8', the network would light up all the neurons which are associated with the edges and loops present in '8', which would further light up the respective neurons in the upcoming layers and so on, till the neuron representing '8' is lit up in the output layer.

To capture any pixel patterns, 'weights' are assigned to each of the connections between the neurons. Then, all the activations are taken and their weighted sum is computed using these assigned weights –

$$w_1a_1 + w_2a_2 + w_3a_3 + \ldots + w_na_n \quad (1)$$

Where:
w = weight
a = activation.

This weighted sum can be any number but for the given network, a number between 0 and 1 is required, that is, a function is required that compresses all the values to values between 0 and 1. This can be done by the 'sigmoid function'-

$$\sigma(x) = 1/1+e^{-x} \quad (2)$$

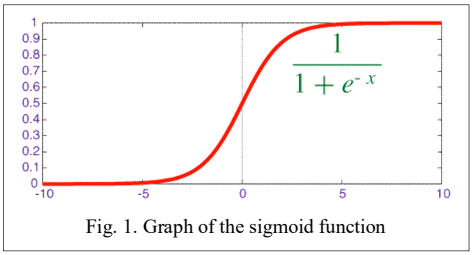

Fig. 1. Graph of the sigmoid function

The weighted sum equation can then be rewritten as –

$$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \ldots + w_na_n) \quad (3)$$

By this equation, it can be said that the activation of the neuron is basically a measure of how positive the weighted sum is.
However, there may be conditions as to when the neuron should light up, that is, it should be lit up when the weighted sum is greater than a certain number b. There should be a 'bias' for it to be inactive. This bias 'b' is included in the equation as follows –

$$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \ldots + w_na_n - b) \quad (4)$$

The 'weights' tell what pixel patterns the neurons in the second layer are picking up on and the 'bias' tells how high the weighted sum needs to be before the neurons start getting meaningfully active.

Every neuron in the second layer is going to be connected to each of 400 neurons in the first layer and each of these connections will have its own assigned weight and bias. On further calculations, it can be said that there would be around 6858 total weights and biases. These 1000+ equations can be represented very easily in a matrix form –

$$\sigma\left(\begin{bmatrix} w_{0,0} & \cdots & w_{0,n} \\ \vdots & \ddots & \vdots \\ w_{k,0} & \cdots & w_{k,n} \end{bmatrix}\begin{bmatrix} a_0^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix}\right) \quad (5)$$

Hence, a neuron is more of a function than a number.

A cost function is a way of telling a system whether the output it has given is useful or not. The cost function adds up the squares of the differences between the incorrect output and the value it is supposed to give, called the cost of a single training example. The average cost over all of the thousands of training examples is considered and this value indicates how useful the network is. When the cost is low, the network is good and when the cost is high, the network is bad.

If the cost function is high, there needs to be a way to let the network know how it can perform better. To explain this further, let there be a function with just one input, rather than the 6858 inputs, and just one number as output and what needs to be found is an input that minimizes the value of the cost function. One way to find the local minimum is to start at any old input and to figure out which direction to move it, greater or lesser, to lower the cost. Hence, for a function with more than one input, the negative gradient of the function can be found, as this gives the direction of the steepest descent, and this process can be repeated till the local minimum is found.

This process of repeatedly moving an input of a function by some multiple of the negative gradient is called gradient descent. It's a way to converge towards some local minimum of the cost function. The initial network is initialized with random weights and biases and then by the gradient descent process, these weights and biases are continuously adjusted. Cost function involves an average over all of the training data, so if it is minimized, it means it's a better performance over all the samples.

The algorithm for computing this gradient efficiently, which is basically the heart of how a neural network works, is called back propagation. This is the algorithm that computes the gradient.

In a stage where the model isn't properly trained yet, the activations will be random and this is what needs to be changed to get the desired output. There are three ways to change the activation - by changing the weight, by changing the bias or by changing the activation from the previous layer. The change in weight is proportional to the change in activation and vice versa. The activation cannot be influenced directly, only the weights and biases can be controlled. Here is where 'propagating backwards' comes in - by adding together all the required changes, a list of changes needed for the second to last layer is created. The same process can be applied recursively to the relevant weights and biases that determine those values and by moving backwards through the network. This same routine of back propagation is followed for every other training example, recording how each example would like to change its weights and biases and then these required changes are averaged together. The stochastic gradient descent is what is used to compute the gradient for the complete dataset.

This is the basics of how an artificial neural network works as the heart of deep learning. There are many more concepts and changes that can be applied to this basic neural

network and that is what is done in the implementation process of text summarization.

VII. Architecture and Types of Artificial Neural Network

There are many variations of Artificial Neural Networks used for different applications. Some of the important ones are listed below –

A. *Convolutional Neural Network (CNN)*

Selects sections of the input and processes them separately in different convolutional layers. It finds its application in areas like image processing.

B. *Recursive Neural Network (RNN)*

This modification is achieved by recursively applying the same weights over the structure to make a prediction by traversing the structure in a topological order.

C. *Recurrent Neural Network*

This variation of an ANN has not only a feed-forward system, but also has a feed-back mechanism, and thus has an element of memory. This is used in applications where the variable changes with respect to time, like in NLP where keeping track of the previous inputs is vital for accurate results.

There are other types like Long Short Term Memory (LSTM) and Sequence-to-Sequence (Seq2Seq) which are more developed versions of the above mentioned basic ANNs.

VIII. Abstractive Summarization

As stated before, abstractive automatic summarization involves understanding and comprehending the source text in a linguistic manner and generating a completely new set of sentences to effectively represent the summary of the source text. Abstractive summarization is mainly of two types, structure based and semantic based. These two types are explained briefly below –

A. *Structure based approach*

This method relies on techniques that use prior knowledge and features like templates and extraction rules. The following are methods of the structure based abstractive text summarization –

1) Tree based approach: A general theme is identified and by a clustering algorithm, the sentences are ordered.

2) Template based method: A template is used that extracts information using fillers and slots.

3) Ontology based method: Ontology represents online documents that are domain connected which have their own information structure. This idea is used to generate new sentences.

4) Graph based method: A graph is created where the vertices denote the words and the edges denote the relation between the words.

5) Lead and body phrase method: This process is used when there is a similarity in the sentence structure where a set of fixed phrases are used in the lead, and body sentences that are used to rewrite the lead sentence.

B. *Semantics based approach*

This method computes the summary by keeping the sentence structure, namely, subject-verb-object as its basis. It also takes into account the sentence position in the entire text and word positions within the sentences. The following are methods of the semantics based abstractive text summarization –

1) Multimodal semantic model: This is used to capture the relation between concepts of multimodal documents.

2) Semantic Graph Model: This method summarizes a document by creating RSG for the initial document by reducing the linguistics graph and then generating the final abstractive outline from the reduced linguistics graph.

3) Information item based method: The information about the summary is generated from abstract representation of supply documents, as opposed to the sentences from supply documents.

4) Semantic Text Representation Model: This technique aims to analyze input text using the semantics of words rather than syntax/structure of text.

IX. Implementation

In our implementation, we summarized a document of text using a combination of both extractive and abstractive methods. First we employed an extractive method to get a smaller section of the data, which we then passed through the abstractive method to generate a new summary.

We started our analysis with a dataset containing articles from February-August 2017 and their summaries. We cleaned our dataset and passed it through an extractive method and stored the extracted summaries. We achieved the extraction by first tokenizing the cleaned data into sentences and creating a similarity matrix having the similarity indices between every two sentences and then converting this matrix into a graph. Similarity between the sentences was found as 1 minus the cosine distance between the two sentence vectors. The TextRank algorithm is applied on the graph to obtain the top n sentences. The concatenation of these top n sentences gave the extracted summary.

The PageRank algorithm is used by Google Search to rank web pages. The idea is that important pages are linked to each other. The PageRank index of a page is essentially the probability of a user visiting it. This can be applied in TextRank by substituting the pages in the PageRank with sentences in the TextRank.

We then passed the extracted summary as the input for the abstractive model, which returned the final abstracted summary.

To achieve this, deep learning using Sequence-to-Sequence (Seq2Seq) ANN was employed. The Seq2Seq model is used to solve any problem which involves sequential information. Text summarization can be modelled as a Many-to-Many Seq2Seq problem. It is often implemented using Recurrent Neural Network (RNN) or Grated Recurrent Neural Network (GRU) or Long Short Term Memory (LSTM) as a memory component is necessary to capture the context of the sentences.

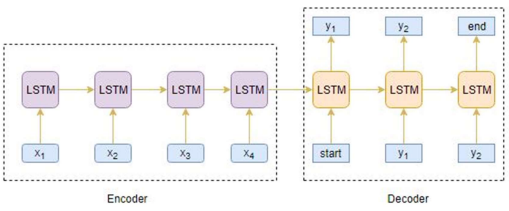

Fig. 2. A typical Seq2Seq model architecture

There are two main components of the Seq2Seq model are, encoder and decoder. Both these parts are of LSTM networks. In the training phase, the encoder reads the entire input sequence and at each timestep, one word is fed into the encoder. It then processes the information and captures the contextual information present in the input sequence.

The decoder reads the entire target sequence word-by-word and predicts the same sequence offset by one timestep. The decoder is trained to predict the next word in the sequence given the previous word. '<start>' and '<end>' are keywords that are added to the start and end of the target sequence. The decoder will start predicting when the first word of the target sequence is passed which is '<start>' and the '<end>' keyword would indicate the end of the target sequence.

After the training phase comes the inference phase where the model is now working on data where the target sequence is unknown. In this phase, the '<start>' keyword is passed to the decoder and the decoder is run for one timestep. The output will be the probability of the next word, so the word with maximum probability is selected. The sampled word is passed as input to the decoder and the internal states are updated with the current timestep. This process is repeated until the '<end>' keyword is reached.

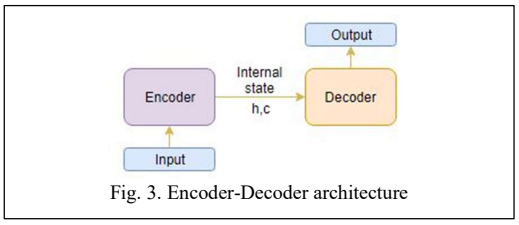

Fig. 3. Encoder-Decoder architecture

We used the above described Seq2Seq model for our implementation.

One of the limitations of this encoder-decoder architecture is that it works only for short sequences because it is difficult for the encoder to memorise long sequences. This problem is fixed with the 'attention mechanism'. This mechanism predicts a word looking only at a few parts of the sequence and not the entire sequence. Instead of looking at each word in the source sequence, the importance of a few specific parts of the source sequence which results in the target sequence. There are two types of attention mechanism – local attention and global attention. In the former, the focus is only on a few source positions and in the latter, the focus is on all source positions.

All the required libraries were imported and a third-party implementation was used to get the attention layer.

Before starting the actual implementation on the dataset, all the repeated and NAN values were dropped and the data was pre-processed. All the unwanted symbols and characters were dropped, all characters were converted to lowercase, contraction mapping was done and all stop words, short words, punctuations and special characters were eliminated. The '<start>' and '<end>' keywords were added to the sentences.

A tokenizer was then built which converted our sentence sequence to an integer sequence. The encoder-decoder LSTM model was then built, whose working has been explained.

The dataset was divided into training data and testing data. The model is trained through multiple epochs of the training data and verified using the testing data. The inference model for the encoder-decoder was then built and then the integer sequence was converted to a word sequence.

The output is the final required summary.

Hence, this is how deep learning is used to achieve text summarization.

X. Conclusion

Text summarization is a growing field with great potential as the necessity for short, condensed abstracts of topics is only increasing with the growing amounts of data on the internet. In this paper, we have reviewed the basics to understand both kinds of text summarization namely, extractive and abstractive. This included brief explanations of multiple approaches to extractive and abstractive methods and an in-detail explanation of neural networks required for the implementation of any kind of abstractive method. We have also briefed over an implementation method for achieving text summarization.

Acknowledgement

This work has been supported by the Computer Science Department of PES University, Bangalore, India. We would like to thank Dr. Uma D. for mentoring us throughout the course of this project and we express our deepest gratitude to her and to the department for providing us with this opportunity.

References

[1] Yogan Jaya Kumar, Ong Sing Goh, Halizah Basiron, Ngo Hea Choon and Puspalata C Suppiah, "A review on automatic text summarization approaches".
[2] Deepali K. Gaikwad and C. Namrata Mahender, "A review paper on text summarization".
[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assef, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, Krys Kochut, "Text summarization techniques – a brief survey".
[4] Alexander M. Rush, Sumit Chopra and James Weston, "A neural attention model for abstractive sentence summarization".
[5] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Calgar Gulcehre and Bing Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond".
[6] Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, Atsushi Otsuka, Hisako Asona, Junji Tomita, Hiroyuki Shindo and Yuji Matsumuto, "Length-controllable abstractive summarization by guiding with summary prototype".

URL References

[1] Attention in Deep Networks with Keras
[2] 7 types of Artificial Neural Networks for Natural Language Processing
[3] Ten trends in Deep learning NLP
[4] Understanding LSTM Networks
[5] Deep Contextualized Word Representations with ELMo
[6] https://www.kaggle.com/sandeepbhogaraju/text-summarization-with-seq2seq-model/data
[7] Comprehensive Guide to Text Summarization using Deep Learning in Python
[8] Introduction to Text Summarization using the TextRank Algorithm
[9] Understand Text Summarization and create your own summarizer in python
[10] https://github.com/aravindpai/How-to-build-own-text-summarizer-using-deep-learning
[11] Shakunni/Extractive-Text-Summarization: Extractive Text Summarization in Python
[12] https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70
[13] https://www.kaggle.com/sunnysai12345/news-summary#news_summary.csv
[14] Comprehensive Guide to Text Summarization using Deep Learning in Python

[15] https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
[16] 20 Applications of Automatic Summarization in the Enterprise
[17] Understand Text Summarization and create your own summarizer in python
[18] (PDF) Study of Abstractive Text Summarization Techniques
[19] Develop a Word-Level Neural Language Model and Use it to Generate Text
[20] How to Get Started with Deep Learning for Natural Language Processing
[21] What's the difference between machine learning and deep learning?
[22] Deep Learning vs. Machine Learning: A Simple Explanation
[23] Difference between Deep Learning & Machine Learning
[24] Understanding Neural Networks: What, How and Why?
[25] Neural Networks and Deep Learning