Journal of Management and Service Science, 2022,
Vol. 02, Iss. 01, S. No. 006, pp. 1-7
ISSN (Online): 2583-1798

Study of Language Models: Evolution & Limitations

Aayush Kumar Gupta[1], Sheenu Rizvi[2]

Amity School of Engineering and technology, Amity University, Lucknow, India[1,2]
aayush8423@gmail.com[1], sheenu_r@hotmail.com[2]

How to cite this paper: A. A. Gupta and S. Rizvi, "Study of Language Models: Evolution & Limitations," Journal of Management and Service Science, Vol. 02, Iss. 01, S. No. 006, pp. 1–7, 2022.

http://doi.org/10.54060/JMSS/002.01.006

Received: 01/01/2022
Accepted: 15/02/2022
Published: 25/03/2022

Abstract

We have come far from the days when rule-based language models used to be the predominant thing in the market. Machine Learning came into play and changed the Language Model industry. In this paper, we will look at how RNN did a much better task for generating output based on its previous results and then how LSTM fulfilled the memory requirement for RNN. Also, we will take a look at how Transformer is much better than RNN combined with LSTM, which is the state-of-the-art language model on which the two best natural processing models like BERT and GPT3.

Keywords

Language Models, Rule-based, Statistical-based, RNN, LSTM, Attention, Transformer

1. Introduction

The language model is a probability distribution over words or words in a sequence. Language models do not interpret the meaning of the words in the sentence as a human does. Language Model predicts the probability of the words that will be likely to appear next. Commonly people misinterpret whenever language models get better than their previous generation to the advent of consciousness. But reaching the level of the singularity isn't that close if the language models can only chat with you by predicting the probability of the next word in the sentence. A trained language model can predict the next word based on the previous word; this is a common model in the Google keyboard. But when we start to add memory to this model things start to get pretty exciting. The model can now predict the probability of the next word by now scanning the previous context making it better. But as the memory is increased the processing gets slower and the output becomes costlier such as GPT-3 has four different models depending upon the cost, speed, and quality of the text processing. So, in summary, language models based on Deep Learning are more of an advanced probabilistic word predicting machine that does not understand the meaning of the words in a grammatical context like a human does.

1.1. Coherence and Long-range Dependencies

All the natural language models require a huge corpus for a dataset. The bigger the corpus for the data set the better the model is trained. But the quality of the output from the language model is determined when the model can be in coherence with its previous context. Let's see how complicated it can get when we try to make our natural language model be in coherence with long-range dependencies. So, whenever the language models try to keep the previous words in the memory it becomes much more computationally expensive. For example, we have a word context of a thousand words, then to predict the next word we have to keep those thousand words in say a probability distribution for the prediction, and to predict the next words it has to keep all the processing of the previous words making the computation to grow exponentially. That is why the earlier language model where memory sequence was the main method for generating the coherence with the words does not seem to work when it has looked back at 100 words thereby losing the overall context.

2. Language Model

Initially, as we step into the Natural Language Processing Models, we can predict the text with the help of supervised learning algorithms where we don't require any kind of structured prediction or sequence prediction. These supervised learning models can be used to do fairly any straightforward task such as classifying whether a document is spam or not.
This also means that we have to take our document as a fixed size vector for the machine to learn. The problem in that case in each document is of variable length and to solve this problem and make it meaningful input to our machine we have converted it into a fixed-size vector.

2.1. Bag of Words

The classic way of solving this problem is the bag of words solution where we have one dimension per word in our vocabulary. In English, vocabulary supposes 100,000 words can be used to create our vector. So, when most of the words apart from the word we cross through will be zero. But this will lead to sparse data and storing zeros will become computationally very expensive. So, we generally store the list of position value tuples of the words or maybe just a list of positions and that becomes a great solution to our computationally expensive problem.

But the key limitation to this problem is that the orders do not matter to this model. In the document, the order of the words is very important for its classification. For example, we have different sentences such as "work to live" and "live to work" both have different separate meanings. But the bag of words model will always score them identically each time because they have the same vectors for the words that are present. All these methods lead to a rise in the "statistical revolution"[1].

2.2. N-gram

Now, the solution to this problem is the N-gram model where you can have bigrams for the pair of two words and then trigrams for pairs of three words. The Dimensionality for these N-grams can be determined by V^N where V is the vocabulary vector and N is the Number of vectors together. So, in an English Trigram, it will require to have 10^15 dimensions which are humongous for any machine to process.

Increasingly, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature (complex-valued embeddings, and neural networks in general have also been proposed, for e.g. speech [2]). Since 2015,[3] the field has thus largely abandoned statistical methods and shifted to neural networks for machine learning [8].

2.3. Recurrent Neural Network

For the solution to the high dimensionality pairs, we have got the Recurrent Neural Networks. The way the RNN solves the previous problem is through an input function that continuously takes the feedback from the output at every stage. In other words, it takes the input from some document and input from itself which will be used to generate the output. It can be visualized as a deep neural network that is processing each input along the way.

The Problem with the normal Recurrent Neural Network is the vanishing and the exploding gradients. As the Recurrent Neural Network is made of many states of hidden layers that help to balance the weight and biases to train the model also imposes the problem of the vanishing gradient.

2.3.1. Recurrent Neural Network

The Hidden Layers are the secret sauce of the neural network. They help the neural network to model complex data with the help of the neurons. We can never know what goes inside the hidden layer. We only supply the input data to the input layer and then the hidden layer does the processing of the data and then maps it to the output layer. Every neural network has at least one hidden layer or it's not a neural network. The neural network that has multiple hidden layers is called the deep neural network. Commonly the hidden layer is fully connected meaning that each neuron in an individual layer is connected to every other neuron in the adjacent two later.

Neurons are the building block of any network. Every neuron is associated with all the neurons from its adjacent previous layer with weights. So, each neuron sums all the weights as per the activation of previous layer neurons and then passes it through an activation function. The activation function is similar to the buffer for each signal to pass. Mostly the activation functions are sigmoid functions where the outputs are limited to the value between zero and one. This way the value of any neuron which sums up the weights as per activation does not either become exponentially large or negative. The sigmoid function thus can give the values that act as the knob for activation of the particular neuron. For example, if the value is closer to one then it means the neurons are actively lit and show greater association to the layer from which the weights are coming, and a similar case follows with the values that are closer to zero. Neural Networks can achieve state-of-the-art results in many natural language tasks, e.g., in language modeling and parsing.[4][5].

2.3.2. Cost Function

The way we train the Recurrent Neural Network is by minimizing the cost function. The cost function is the mean of the squared difference between our actual output and the predicted output. The cost function is the method through which we can say to our neural network that it has generated a bad result. If the generated output is away from the actual output then the cost function will be higher. And if the generated output is closer to our actual output it will be minimum. As the cost function is algebraic, therefore we can global minima of the function with the help of gradient descent [7].

2.3.3. Gradient Descent

The gradient descent is the method through which we find the slope of the function at any given random value. For example, there is a very simple function.

Example, $ax2 + by2$

Then we can put any random values of x and y and from there we can determine the slope of that point on the function. If the slope that we determined at the random location (x, y) comes out to be positive then the gradient descent will move

steps downhill and if the slope of the random location comes out to be negative then the gradient descent will move steps uphill. We can consider the gradient descent function for even three dimensions where we can plot two dimensions as input and one dimension as output which will be more realistic to the Recurrent Neural Networks where two inputs are given to the neural network and one output is generated.

Now back to our gradient descent we have seen how it is determined to go uphill or downhill depending upon the slope generated at random steps. When we plot the graph of gradient descent it will seem to have a ball-like motion. It is because gradient descent is trying to reduce the number of steps taken. The gradient descent takes huge steps when the slope is steep and small steps when the slope is flat. Because when the slope is steep it is far away from the local minima and when the slope is flat it seems to approach the local minima and thereby reducing the cost function. So, the gradient descent will carefully take small steps whenever it starts to reach the local minima of the cost function, that is how it will reduce the cost function and thereby train the complete network for better results [6].

The gradient is calculated for the complete input batch at once therefore whenever the inputs are fed into the function of the network, the network will then go to minimize the cost by analyzing then reiterating every time until the predicted output is the same as the actual output.

2.3.4. Stochastic Gradient Descent

As we see in the gradient descent, the gradient will only be calculated once it has processed all the inputs only. Then the gradient will then be used for comparison between the predicted and the actual value and the difference will determine the balancing of the weights and the biases.

But in the stochastic gradient descent, the gradient will be calculated at every input. Therefore, as the gradient is required for calculating the cost function and then balancing the weights in the network that is why stochastic gradient descent will provide us with faster results as gradient descent is calculated at every input.

2.3.4. Backpropagation

As we know that our neural network is made of layers of neurons. We have already seen that the neural network is made of many hidden layers between the input layer and the output layer. Each layer is made of many neurons. These neurons get activated depending upon the weights & biases that the previous hidden or input layer carries and also if the previous neuron is activated or not.

The main work of backpropagation is adjusting the weights and the biases of each neuron depending upon the output layer result and the cost function. When it has been determined that the cost function is not up to the mark it will nudge all the weights and biases of the neuron within the layers of the network. When the pattern is formed between the input layer, hidden layers, and the output layer then the backpropagation will provide the biggest increase to the weight or biggest strengthening of connections with the neurons that are most active and the neurons that we wish to become more active. Backpropagation is analogous to the neuroscience Hebbian Theory which states that “Neurons that fire together, wires together.”

Therefore, the backpropagation can influence the neural network either through weights or biases. For example, while determining the next word depending upon the previous word, the neuron containing the previous word which is closely related to the next word will increase the weight associated with the particular neuron will increase and this will repeat to all of the previous layers. As the activation of the neuron is determined by sigmoid function therefore it will also result in a greater activation value of the neurons.

Therefore, the neurons are nudged depending upon the proportion of the corresponding weights and in proportion to

how much those neurons need to change and this what the idea behind propagating backward. By adding all the desired results, we get the list of all the nudges that we want to happen from the second to the last layer.

2.4. Long short-term memory

The problem with the Recurrent Neural Network was that in multiple layered neural networks when the activation function such as sigmoid was used they squished the inputs it received to only 0 and 1. This made it hard for the neural network to learn.

As it makes it more difficult to adjust the weight during the time of backpropagation as when the activation function output value is near zero or zero then adding or multiplying the learning rate will have negligible to no effect.

There is a great analogy to this problem: suppose student A is not able to attend the school for a week and student A asks his friend student B to keep him updated. As student B is busy after classes, student B transfers his learning (70 percent of what was originally taught) to student C (a mutual friend of student A and student B) to transfer it to student A. Now student C transfers his 70 percent of retention of the learning to student A. Now, student A learning becomes (70 percent of student C * (70 percent of student B)) = 0.49 of the original teaching. This is similar to how the backpropagation gradient vanishes due to the sigmoid squashing effect.

There were some solutions for solving the vanishing gradient problem such as ResNet and ReLu. But LSTM (Long Short-Term Memory) showed the greatest enhancement to the RNN vanishing gradient problem.

The LSTM solved the problem of the vanishing gradient by adding some matrix multiplier to the activation function. The LSTM vector has two more hidden states that are constantly learning. Instead of the network remembering all the words, the LSTM creates a vector that is the result of the input of the previous vector and the new word. This is how LSTM was solving the vanishing gradient problem.

The Limitation of the LSTM was that it was difficult to train because the LSTM has a very long gradient path, for example, 100-word text can have a 100-layer network. The training for the LSTM network was serial as it requires the vector from the previous LSTM cell and word to be trained.

2.5. Transformer

The latest revolution in the natural language processing space after the Recurrent Neural Network was the transformers. To date, the transformer is predominant with various other variants being added to it. Like two most trained transformers exist and are BERT (Bidirectional Encoder Representations and Transformers) and GPT3 (Generative Pre-Trained Transformer 3). These both are the state-of-the-art natural language processing model containing billions of parameters for natural language processing. There are even AI companions that have been developed such as Replica which can produce a human conversation that is based on the GPT3 Model by OpenAI.

2.6. Attention

The Transformer gained popularity when it was mentioned in the paper “Attention is what you need”. As described in the paper, the attention model worked with a query, key, and relevancy vector. For every output, we are considering a query vector and for every input, we are considering a key vector and the relevance score is the dot product of those vectors. And then we normalize the output vector by the softmax function. The relevance of the output words is calculated by the product of the query and the key.

In the attention model, we use the key and query token as the input, and then we produce these vectors which then

result in the relevancy of the output. Now to generate the output for the next layer we simply multiply the softmax of the relevancy vector by the value that we receive from the previous layer of the network.

2.6.1. Multi-headed Attention

Multi-headed attention is the architecture that is used in the transformers to learn different semantic meanings for the same context. We have to repeat the same process of finding the relevance and the output of the query and the key vector with different values of the query, key, and value vector that we have passed the previous time.

So, in the context of translation, we can create different multi-head attention models for grammar, vocabulary, etc. The model can look in the document for different purposes which makes this model much more flexible for processing sequences of corpus text.

2.7. Potential Encoding

Without the positional encoding, the transformer attention model is the bag of word models. In the input layer of the attention model, the input layer is created by adding the word embedding (e.g., word2vec) and a position embedding which then can be passed to a multi-headed attention model. This lets the model reason the relative position of any token. That is the reason the attention model differs from the bag of word models.

2.8. Transformer and Recurrent Neural Network

The key advantage of transformers is that they are completely parallel. This means that with the help of a high spec GPU we can perform these operations literally at a pace that we had not imagined earlier. As in the case with RNN, we cannot do the parallel operation because if one token is busy operating then we have to require the same token for the next processing phase. That is why RNN, despite being a revolution in natural language processing, did not succeed as the convolutional neural network in computer vision.

Also, the other problem which we faced in RNN was with the activation function that was sigmoid and tanh activation. It means that the neural network can only be trained with binary input in the neural network layer. This also means that we can be stuck in the problem of inefficient learning with backpropagation. The transformer utilized the ReLu activation function, which is a similar activation to sigmoid but its value can be scaled to infinity which means that now the activation function can express itself on the scale of zero to infinity rather than being binary values.

Conclusion

Natural Language Processing has come a long way from a simple text translation George time experiment in 1954 to the current where the development of Deep Neural networks is capable of making autonomous decisions. We have gone through how Natural Language Processing evolved from just a rule-based method to utilizing statistics using machines and then developing Neural Networks for training itself on the labeled data to produce state of the art results.

The Natural language Processing algorithms have also gone through regress improvement. We have looked at how statistical algorithms such as a bag of words and N-gram have been substituted by the word2vec algorithm. And now with the development of neural networks, newer algorithms are replacing and modifying the old ones such as gradient descent has been replaced by stochastic gradient descent, and also the inclusion of LSTM to work with the RNN networks for better memory. The significant leap in the Natural Language model came when the Neural Network started using the Transformer architecture from the paper “Attention is all you need”. This paper is the hallmark of this decade in the NLP field.

After the Transformer came into the picture many giants like Google, Facebook, openAI, and many more started their

model development resulting in many NLP Transformer models currently used in industry such as BERT, GPT, XLNET, ALBERT, and many more. Now the Natural Language Processing models are heading toward cognition to emulate the intelligent behavioral pattern and comprehensions in the NLP. More work is put into developing the cognitive ability of machines that can parallel the human mind. It will be interesting to observe what the Natural Language Processing Model holds for the future.

References

[1] M. Johnson, “How the statistical revolution changes (computational) linguistics,” in Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics Virtuous, Vicious or Vacuous? - ILCL ’09, 2009.

[2] S. Kumar et al., “Novel method for safeguarding personal health record in cloud connection using deep learning models,” Comput. Intell. Neurosci., vol. 2022, no. 3564436, pp. 1-14, 2022.

[3] M. Trabelsi, P. Kakosimos and H. Komurcugil, "Mitigation of grid voltage disturbances using quasi-Z-source based dynamic voltage restorer," 2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018), 2018, pp. 1-6, doi: 10.1109/CPE.2018.8372574.

[4] S. Kumar, P. K. Srivastava, G. K. Srivastava, P. Singhal, D. Singh, and D. Goyal, “Chaos based image encryption security in cloud computing,” J. Discrete Math. Sci. Cryptogr., pp. 1–11, 2022.

[5] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, “Exploring the limits of Language Modeling,” arXiv [cs.CL], 2016.

[6] S. Kumar et al., “Protecting location privacy in cloud services,” J. Discrete Math. Sci. Cryptogr., pp. 1–10, 2022.

[7] D. K. Choe and E. Charniak, “Parsing as language modeling,” in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016.

[8] Iop.org. [Online]. Available: https://iopscience.iop.org/article/10.1149/10701.15533ecst/meta. [Accessed: 23-Jul-2022].