Neural network for Language Modelling

CS6081
Lecturer: Jim Buckley

Student ID: 18043038
Name: Shravan Chandrashekharaiah

Table Of Content

Abstract

The neural network is a complex algorithm to predict the unknown data from the previously trained model. The recurrent neural network is an advanced neural network, wherein the memory is introduced to maintain the context of the previous data and predict the unknown data. The recurrent neural network is ideal where sequential data is analyzed. This paper consists of applying neural networks and RNN for language modeling techniques. Language modeling is the process of modeling an algorithm to train the set of data and predict the output based on the sequential data that is fed. Language modeling is central to many important natural language processing tasks. Analysis of RNN towards Language modeling and its limitations is covered.

Introduction

Computer languages, like programming languages, can be specified completely. Complete set of keywords or the reserve words can be predefined and all the ways that are valid that can be used can be accurately defined.

But it is not true in the case of Natural Language. Natural language is random, it is not designed in one particular way. It will evolve and hence there is no such fixed formal definition of the usage of the natural language.

There may be formal rules for parts of the language, and heuristics, but the natural language that does not confirm is often used. Natural language will be having a huge number of terms that are used in many different ways which will bring in ambiguities, yet still can be understood by other humans. Further, languages change, word usages change, it is a moving target.

Nevertheless, linguists try to specify the language with formal grammars and structures. It can be done, but it is very difficult and the results can be fragile. An alternative approach to specifying the model of the language is to learn it from examples.

Neural Network

Neural Networks are a collection of algorithms that are loosely modeled on the human brain to identify the pattern. Neural networks help us to classify and cluster data. Recurrent neural networks(RNN) are the state-of-the-art neural network algorithm designed to recognize patterns in sequence data such as text, genomes, handwriting, stock markets, etc.

RNNs are stable and convincing types of neural networks for the analysis of sequence data and they make a very strong statement because they are one of a kind in terms of inner memory neural networks. Like any other human being, as you are reading an essay, you won't start learning from scratch, you'll be able to correlate terms to previous ones and get the sense out of

it. People's thoughts are resilience. This can not be done by current neural networks, and it is a big limitation. This issue is addressed by RNN. There are networks that have loops in them, which cause the information to persist.

Like any other deep learning algorithm, the recurrent neural network is relatively old. Its origins date back to the 1980s, but its significance was brought to light only in recent years. RNN took the driving seat in deep learning algorithms with significant advancements in computing hardware and the development of long-term memory (LSTM) in the 1990s.

It can remember the important things about the information they got, which makes their prediction very accurate, as RNN's have internal memory. Compared to other neural networks, RNN can obtain a much deeper understanding of a sequence and its meaning.

Why Recurrent Neural Network

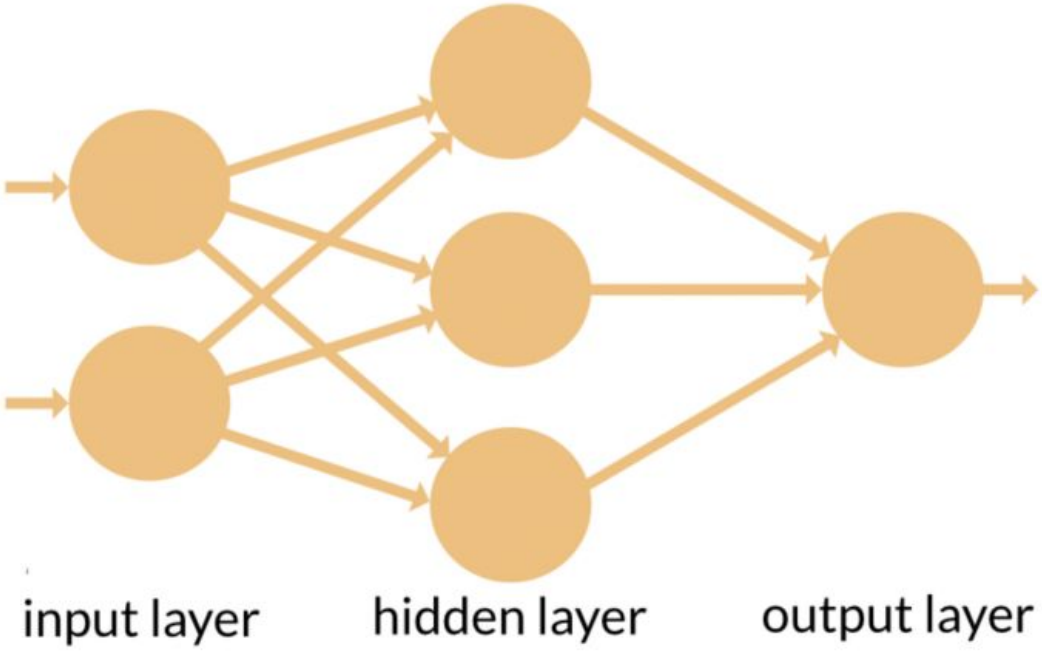
Since we can conclude from above that the neural network only focuses on the current new input and gives priority to the input context, the sequential information can not be treated. It has no internal memory and can not recall previous data and in such situations, sequence information is therefore not valid. Recurrent neural networks are required to address this issue without remembering the context.

How Recurrent Neural Networks work

The recurrent neural network is better understood when opposed to the working sequence data of the feed-forward neural network.

Essentially, sequence information is only structured data in which similar objects obey one another. Types have stock market data or the sequence of DNA. Perhaps the most popular types of sequential data are time-series data, which is just a series of time-ordered data points.

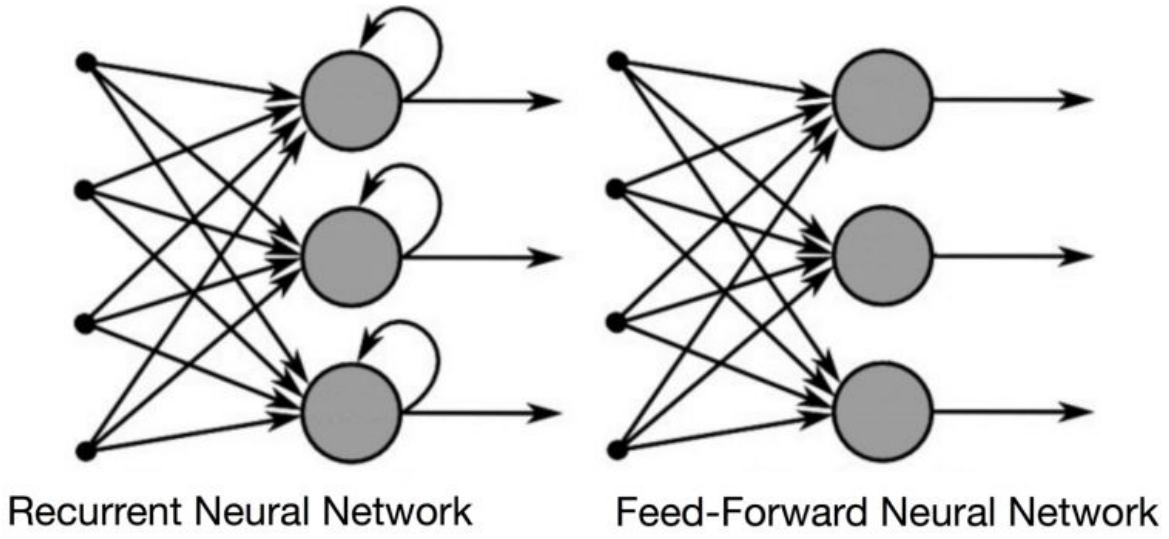
RNN vs Feed-Forward Neural Networks



Neural networks of RNN and feed-forward derive their names from the way they channel data. The knowledge only travels in one direction in a feed-forward neural network— from the input layer, through the hidden layers to the output layer. The data passes through the network straight away and never reaches a node twice.

Feed-forward neural networks have no knowledge of their data and are poor at predicting what will happen next. Since a feed-forward network only considers the latest input, it does not have a timely notion of order. It actually can't remember anything except its learning about what happened in the past. Using a chain, data loops in RNN. In making a decision, it takes into account the latest feedback as well as what it has learned from the previously obtained inputs.

The two images below show the difference between an RNN and a feed-forward neural network in the flow of information. A common RNN has a memory for the short term. These also have a long-term memory in tandem with an LSTM.



RNN vs FFNN [10]

Imagine having a standard neural feed-forward network and sending it the word "network" as an input and process character-by-character word character. By the time it reaches the character "w" it has already forgotten about "n," "e" and "t," which makes predicting which character will come next almost impossible for this form of neural network. Nevertheless, due to its internal memory, a recurrent neural network is able to remember these characters.

An RNN thus has two inputs: the past of the present and the recent past. This is important because the data sequence contains crucial information about what is to come next, which is why an RNN is unable to do other algorithms.

As with all other deep learning algorithms, a feed-forward neural network assigns a weight matrix to its inputs and then generates the output. In addition, the weights will also be modified by a recurrent neural network by gradient descent and time backpropagation (BPTT).

Two issues of Standard RNN's

A gradient in terms of its outputs is a partial derivative. Another way to put it as a gradient calculates how much a function's output changes if the input is changed a bit.

The gradient can also be considered as a function's slope. The steeper the slope, the higher the gradient, and the faster a model can learn. But the model stops learning if the slope is null. A gradient essentially calculates the change with respect to the difference in error in all weights.

Exploding Gradients

Exploding gradients are when the weights are given stupidly high importance by the algorithm, without much justification. Fortunately, truncating or squashing the gradients will easily solve this problem.

Vanishing Gradients

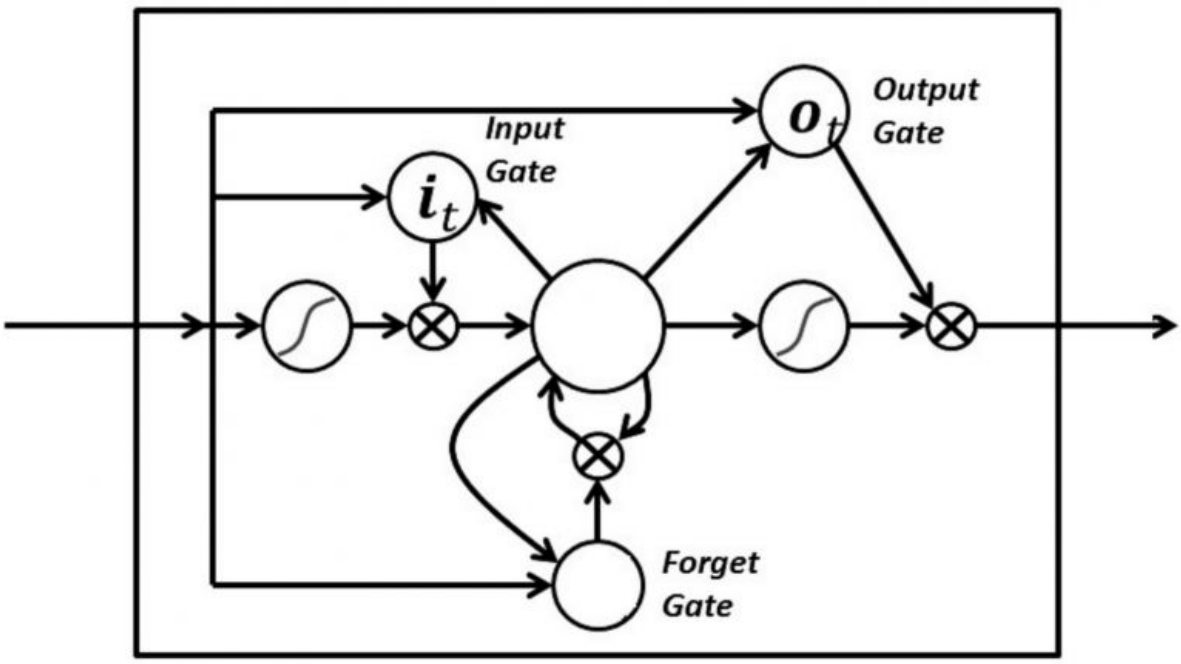
Vanishing gradients occur when a gradient's values are too small, resulting in the model stopped learning or taking much too long. In the 1990s, this was a major problem and much harder to solve than the gradients which exploded. Luckily, Sepp Hochreiter and Juergen Schmidhuber solved it through the LSTM method.

Long Short-Term Memory(LSTM)

Long-term memory networks are an extension of recurrent neural networks, basically expanding the memory. Hence, learning from essential interactions that have long lags in between is well adapted.

LSTMs allow RNNs to keep track of inputs for a long time. This is because, like a computer's memory, LSTMs contain information in a database. The LSTM is able to read, write, and delete memory information.

There are three gates in an LSTM: the gate of entry, forgetting and output. These gates decide whether to allow new input (input gate) or not, remove the data because it is not necessary (forget gate) or allow it to influence the output at the current time (output gate). Below is an example of the three gates of an RNN[9]



Three gates of an RNN[9]

Language Modeling

Statistical Language Modeling

Statistical Language Modeling, or Language Modeling and LM for short, is the creation of probabilistic models that, given the words that precede it, can predict the next word in the string[1].

On the basis of text instances, a language model learns the likelihood of word occurrence. Simpler models can look at a short sequence of words, while larger models can work at the sentence or paragraph level. Most commonly, word-level language models operate[2].

For a wide range of natural language processing functions, language modeling is a root problem. More practically speaking, language models are used for a project that requires language comprehension on the front-end or back-end of a more complex model[1].

Likewise, in many related natural language processing tasks, language models are used to produce text, such as:

Optical Character Recognition.
Handwriting Recognition.

Machine Translation.
Spelling Correction.
Image Captioning.

Language modeling is the art of determining the probability of a sequence of words. This is useful in a large variety of areas including speech recognition, optical character recognition, handwriting recognition, machine translation, and spelling correction[3]. Models that perform better on their planned natural language processing tasks also result in the development of better language models.

Neural Language Models

Recently, the use of neural networks has become very common in the design of language models, to the point that it may now be the preferred approach. Neural networks are often referred to as Neural Language Modeling, or NLM for short, in language modeling[9].

Neural network approaches achieve better results than conventional methods on isolated language models as well as when systems are incorporated into larger models on challenging tasks such as speech recognition and machine translation. The ability of the approach to generalizing may be a key reason for the leaps in improved performance[1][109].

In particular, a word embedding is implemented using a real-valued vector to represent that term in a vector space of a project. A learned representation of words based on their use makes a similar representation of words with a similar meaning[4]. Such generalization is something that can not easily be achieved by the representation used in conventional statistical language models[5].

The approach of the neural network to language modeling can be represented using the following three model properties[6]:

- Associate every word with a distributed word function vector in the vocabulary.
- In terms of the feature vectors of these words in the sequence, describe the combined probability function of word sequences.
-
- Learn the word feature vector and the probability function parameters at the same time.

This is a relatively simple system in which both the representation and probabilistic models are learned directly from raw text data together[7][8].

The neural-based methods have recently begun to outperform the traditional numerical approaches and then steadily continued to do so.

The implementation of the approach was initially based on feed-forward neural network models. RNN was then adopted.

Research question

Can LSTM be used to get a more accurate prediction for language modeling?

Pilot Study

In order to address the research question, text generation analysis will be done. Text Generation is one such task that can be architectured using deep learning models, particularly Recurrent Neural Networks. Text Generation is a type of Language Modelling problem. Language Modelling is the core problem for a number of natural language processing tasks such as speech to text, conversational system, and text summarization. A trained language model learns the likelihood of occurrence of a word based on the previous sequence of words used in the text. Language models can be operated at the character level, n-gram level, sentence level or even paragraph level.

For the analysis for an LSTM over RNN, data is collected and split into training data and test data. Model for both RNN and LSTM is constructed. Both the models are trained for training data and then prediction accuracy is measured for both the models.

For the analysis, we will be using python language and Keras library for the prediction of LSTM and RNN prediction. And will be modeling LSTM as below in three layers:

- Input Layer: Takes the sequence of words as input
- LSTM Layer: Computes the output using LSTM units. Adding 100 units in the layer, but this number can be fine-tuned later.
- Dropout Layer: A regularisation layer which randomly turns-off the activation of some neurons in the LSTM layer. It helps in preventing overfitting.
- Output Layer: Computes the probability of the best possible next word as output

From the above modeling, we will run the accuracy-test for LSTM and then compare it with the RNN and conclude the research with the accuracy of text generation, a language modeling application, is better with RNN or LSTM.

References:

1. Yoav Goldberg, Page 105, 109 - Neural Network Methods in Natural Language Processing (2017)

2. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, Page 238, Introduction to Information Retrieval

4. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush, - Character-Aware Neural Language Models, 2015

5. Holger Schwenk, Jean-Luc Gauvain - Connectionist language modeling for large vocabulary continuous speech recognition, 2002

6. Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin - A Neural Probabilistic Language Model, 2001

7. Toma´s Mikolov, Martin Karafiat, Lukas Burget, Jan “Honza” Cernock, Sanjeev Khudanpur2 - Recurrent neural network based language model, 2010

8. Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černocký, Sanjeev Khudanpur - Extensions of recurrent neural network language model - 2011

9. Yoshua Bengio, Rejean Ducharme and Pascal Vincent - A Neural Probabilistic Language Model - 2003

10. Goyal P., Pandey S., Jain K. (2018) Unfolding Recurrent Neural Networks. In: Deep Learning for Natural Language Processing. Apress, Berkeley, CA