BYTE QUEST

Vasavi College of Engineering

Department of Computer Science and Engineering

October 30, 2019

Volume 75

Contents:

Byte Quest is the article published by the CSE dept of Vasavi College of Engineering regarding the latest innovative Technologies and Software that have been emerged in the competitive world. The motto of this article is to update the people regarding the improvement in technology. The article is designed by the active participation of students under the guidance of faculty coordinators.

 Good, bad or indifferent if you are not investing in new technology, you are going to be left behind.

-Philip Green

 Once a new technology rolls over you, if you're not part of the steamroller, you're part of the road.

-Stewart Brand

FACULTY CO-ORDINATORS

M.SUNDARI (ASST. PROFESSOR)

GARIMA JAIN (ASST. PROFESSOR)

STUDENT COORDINATORS

ESHWAR (4/4 CSE-A)

G CAROL (3/4 CSE-A)

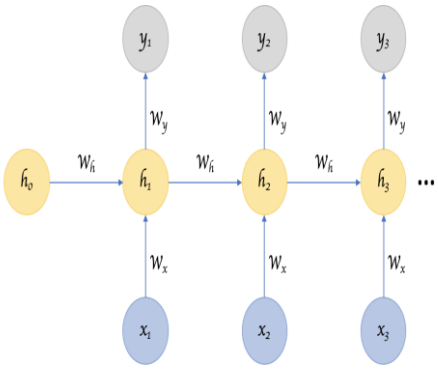
ABHINAV (2/4 CSE-A)

SREEEJA (4/4 CSE-B)

D. APARNA (3/4 CSE-B)

ANISHA (2/4 CSE-B)

RECURSIVE NEURAL NETWORK

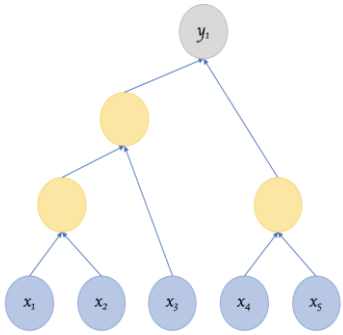


Recurrent Neural Networks (RNNs) add an interesting twist to basic neural networks. A vanilla neural network takes in a fixed size vector as input which limits its usage in situations that involve a 'series' type input with no predetermined size. RNN remembers the past and it's decisions are influenced by what it has learnt from the past.

Recurrent Neural Networks remember the past. For example, an image classifier learns what a "1" looks like during training and then uses that knowledge to classify things in production. While RNNs learn similarly while training, in addition, they remember things learnt from prior input(s) while generating output(s). It's part of the network. RNNs can take one or more input vectors and produce one or more output vectors and the output(s) are influenced not just by weights applied on inputs like a regular NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s). So, the same input could produce a different output depending on previous inputs in the series

MANVITH REDDY (CSE-B 2/4)

RECURSIVE NEURAL NETWORK

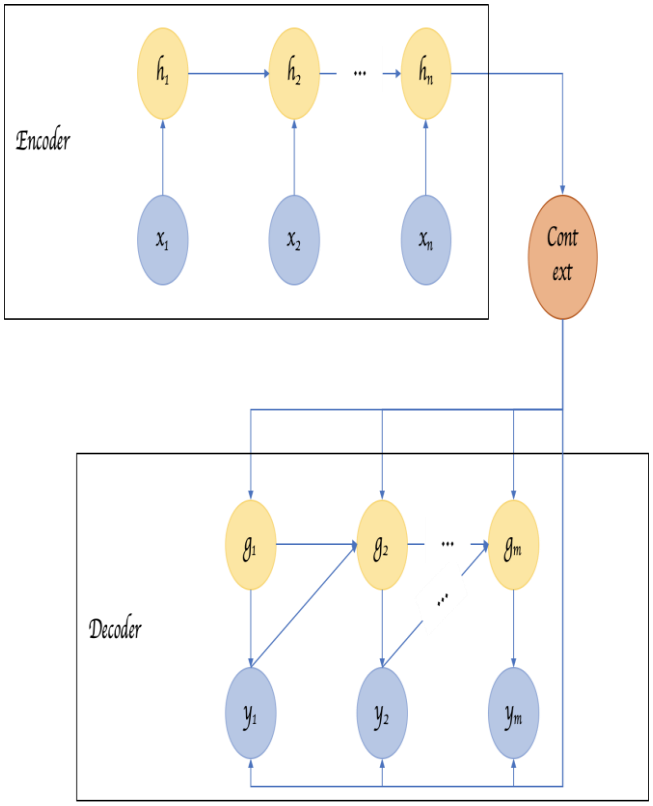


A recurrent neural network parses the inputs in a sequential fashion. A recursive neural network is similar to the extent that the transitions are repeatedly applied to inputs, but not necessarily in a sequential fashion. Recursive Neural Networks are a more general form of Recurrent Neural Networks. It can operate on any hierarchical tree structure. Parsing through input nodes, combining child nodes into parent nodes and

Parsing through input nodes, combining child nodes into parent nodes and combining them with other child/parent nodes to create a tree like structure Recurrent Neural Networks do the same, but the structure there is strictly linear. i.e. weights are applied on the first input node, then the second, third and so on. But this raises questions pertaining to the structure. If the structure is fixed like in Recurrent Neural Networks then the process of training, backprop etc makes sense in that they are similar to a regular neural network. But given the complexity involved, both computationally and even more importantly, in getting the requisite training data, recursive neural networks seem to be lagging in popularity to their recurrent cousin.

NAEHAL (CSE-B 2/4)

ENCODER DECODER SEQUENCE TO SEQUENCE RNN



Encoder Decoder or Sequence to Sequence RNNs are used a lot in translation services. The basic idea is that there are two RNNs, one an encoder that keeps updating its hidden state and produces a final single "Context" output. This is then fed to the decoder, which translates this context to a sequence of outputs. Another key difference in this arrangement is that the length of the input sequence and the length of the output sequence need not necessarily be the same. Sometimes it's not just about learning from the past to predict the future, but we also need to look into the future to fix the past. In speech recognition and handwriting recognition tasks, where there could be considerable ambiguity given just one part of the input, we often need to know what's coming next to better understand the context and detect the present.

This does introduce the obvious challenge of how much into the future we need to look into, because if we have to wait to see all inputs then the entire operation will become costly. And in cases like speech recognition, waiting till an entire sentence is spoken might make for a less compelling use case. Whereas for NLP tasks, where the inputs tend to be available, we can likely consider entire sentences all at once. Also, depending on the application, if the sensitivity to immediate and closer neighbors is higher than inputs that come further away, a variant that looks only into a limited future/past can be modeled. We cannot close any post that tries to look at what RNNs and related architectures are without mentioning LSTMs. This is not a different variant of RNN architecture, but rather it introduces changes to how we compute outputs and hidden state using the inputs. In a vanilla RNN, the input and the hidden state are simply passed through a single tanh layer. LSTM (Long Short Term Memory) networks improve on this simple transformation and introduces additional gates and a cell state, such that it fundamentally addresses the problem of keeping or resetting context, across sentences and regardless of the distance between such resets. There are variants of LSTMs including GRUs that utilize the gates in different manners to address the problem of long term dependencies. This is then fed to the decoder, which translates this context to a sequence of outputs. we often need to know what's coming next to better understand the context and detect the present

MOHAMMED AZAM (CSE-B 2/4)