Chapter 6: Deep learning for text and sequences

Just as deep learning can be used on images, it can also be used on text, sequences and timeseries data. The two deep learning algorithm use in handling text and sequence are the Recurrent neural network (RNN) and the 1D convnets. Before text,timeseries data can be feed into any of the network they should be preprocessed. For time series data,they should be normalize while text data should be vectorized.Vectorizing text is the process of transforming text into numeric tensors. This can be done in multiple ways:
1) Segment text into words, and transform each word into a vector.
2)Segment text into characters, and transform each character into a vector.
3) Extract n-grams of words or characters, and transform each n-gram into a vector.
N-grams are overlapping groups of multiple consecutive words or characters.

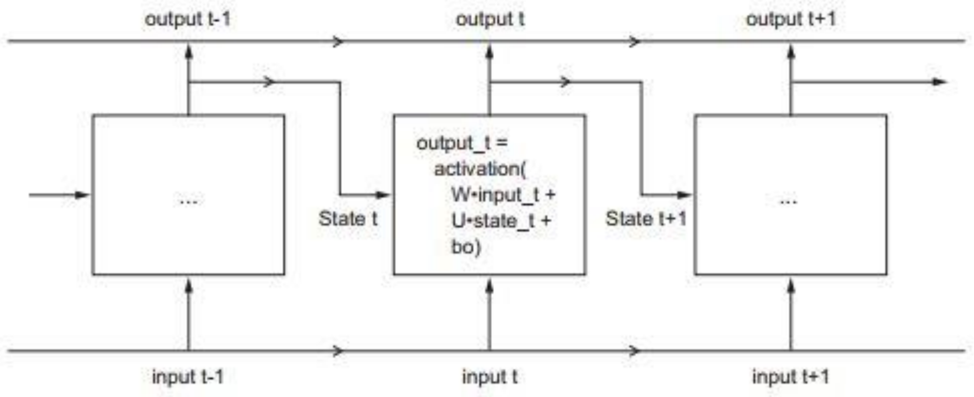
Words, characters and n-gram are called token and breaking text into tokens are called tokenization. two major ways of turning vectors into tokens are 1) One hot encoding 2) token embedding (Word embedding)

One hot encoding: is a way to turn tokens into a binary vector of n dimension where n is the size of the vocabulary. *the vector is all zeros except for the* i *th entry, which is 1.*When the token ina vocabulary become so large a variant of one-hot encoding called one-hot hash trick.

Word embedding: Is another way to represent words with a low dimensional floating point vector. Word Embedding help in getting to the heart of the meaning of text, and the semantic relationship between tokens. Word embedding are learned from the data by using a pretrained word embedding model or by learning it with the task at hand.

After vectorizing a sequence and text data the next step is to feed it into deep learing algorithm to be used

.1) Recurrent Neural Network (RNN): it processes sequences by iterating through the sequence elements and maintaining a *state* containing information relative to what it has seen so far It uses the concept of looping and state to keep up with the sequence. *It loops over timesteps, and at each timestep, it considers its current state at* t *and the input at* t *of shape* (input_ features,), *and combines them to obtain the output at* t*. You'll then set the state for the next step to be this previous output. For the first timestep, the previous output isn't defined; hence, there is no current state but set to zero vector.*



Simple Recurrent Neural Network unrolled over time.

The problem with simple recurrent network is the issue of vanishing gradient. This make it not remember information after many timesteps. Two ways this problem has been solve is with Long-short Term Memory LSTM and Gated recurrent Units GRU.

Long short Term Memory: This help to solve the long term dependencies, by having another state in the hidden layers called cells used for storing relevant information over time. The cells have three gate, input gate, forget gate and output gate. These gates control the flow of information which is needed to predict the output in the network. Detailed working of the gate is explained in this link https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The overfitting problem in recurrent neural network can be solved using 1) Recurrent dropout 2) stacking recurrent layers 3) bidirectional recurrent layers