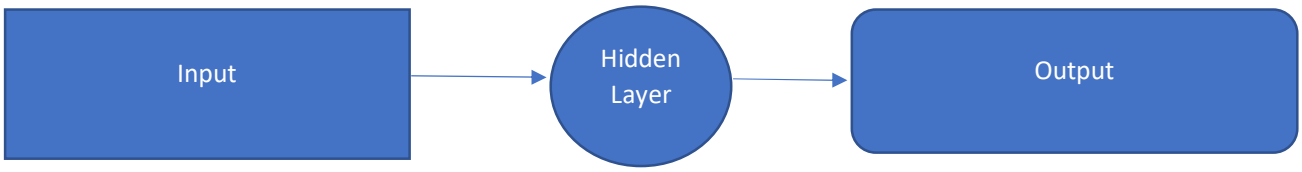
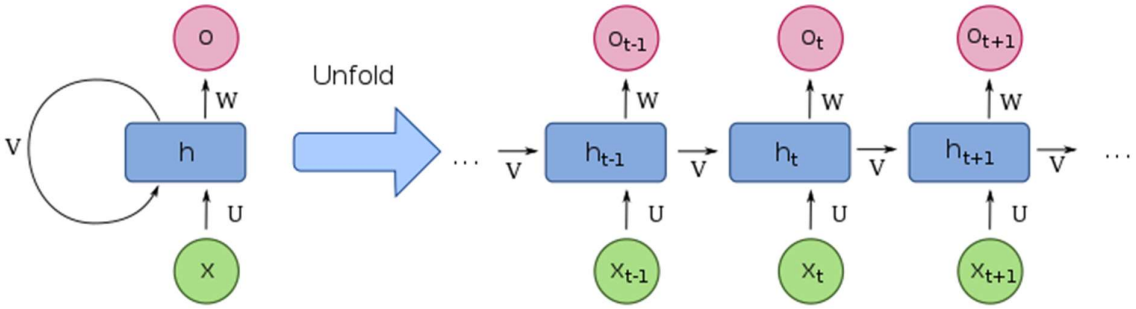
Recurrent Neural Networks (RNN)

Recurrent neural networks (RNNs) are a type of neural network in which the results of one step are fed into the next step's computations. Traditional neural networks have inputs and outputs that are independent of one another, but there is a need to remember the previous words in situations where it is necessary to anticipate the next word in a sentence. As a result, RNN was developed, which utilised a Hidden Layer to resolve this problem. The Hidden state, which retains some information about a sequence, is the primary and most significant characteristic of RNNs.



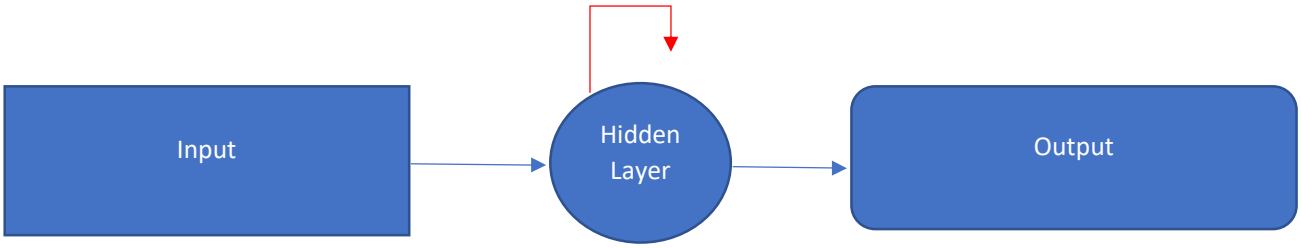
RNNs have a "memory" that retains all data related to calculations. It executes the same action on all the inputs or hidden layers to produce the output, using the same settings for each input. In contrast to other neural networks, this minimises the complexity of the parameter set.

Consider a deeper network that has three hidden layers, one output layer, one input layer, and three hidden levels. Each hidden layer will thus, like other neural networks, have its own set of weights and biases. For example, let us suppose that the weights and biases for hidden layer 1 are (ht-1), (ht) for the second hidden layer, and (ht+1) for the third hidden layer. This indicates that each of these layers is independent of the others and does not retain information from earlier outputs.



By giving all of the layers the same weights and biases, RNN transforms independent activations into dependent activations, decreasing the complexity of raising parameters and memorising each previous output by using each output as an input to the following hidden layer.

Thus, all three layers can be combined into a single recurrent layer so that the weights and bias of all the hidden levels are the identical.



Since we are told to do some form of natural language processing for RNN, we have two options, either Text mining or Sentiment analysis.

Advantages of Recurrent Neural Network

1. An RNN retains every piece of knowledge throughout time. Only the ability to remember past inputs makes it helpful for time series prediction. Long Short-Term Memory is the term for this.
2. Convolutional layers and recurrent neural networks are even combined to increase the effective pixel neighbourhood.

Disadvantages of Recurrent Neural Network

1. Problems with gradient disappearing and explosions.
2. It is exceedingly tough to train an RNN.
3. If tanh or relu are used as the activation function, it cannot process very long sequences.

Reasons why the chosen dataset is appropriate for Recurrent Neural Networks

Dataset: https://www.kaggle.com/code/slythe/twitter-sentiment-analysis-custom-model/data

1. Sequential data is where RNN shines the most. Any input or output length is supported. RNN processes arbitrary input sequences using its internal memory. As we can see in our dataset of Twitter reviews, it is sequential data.

2. RNNs is the most effective in predicting the following words in a string of words. Like the human brain, more importance is placed on recent information to anticipate sentences in dialogues. The current dataset is suitable as it contains strings of words.

3. The information checks all the boxes:

   a. Variety
   b. Quality – no empty sets
   c. Quantity – 500 000 entries

4. Regarding NPL the data must contain corpus, corpus represents a collecting of texts, deriving from the Latin word which means “body”. We can clearly see that our data has corpus.

Analysis that will be performed on the dataset

(Refer to the jupyter notebook for much more detail on the analysis process)

1. The dataset that we will be using is a sentiment dataset, which contains 1.6 million entries from Twitter using the twitter api. The Tweets have been annotated (0 = negative, 2 = neural and 4 = positive) and by using this we can detect sentiments.

   We will be using this data to build a sentiment detection algorithm.

2. Libraries we will import:
   a. Pandas
   b. Sklearn
   c. TensorFlow
   d. Numpy
   e. Seaborn
   f. Matplotlib
   g. Nltk

3. The dataset columns have no names, so we will give them appropriate names.

4. Explore and visualize the data, to get a better idea of what is happening and how to use our data.

5. Text Pre Processing - The fact that all our data is in text format is our major problem (strings). The classification task will be carried out by the classification algorithms that we have learned about so far using numerical feature vector. A corpus can be converted to a vector format using a variety of techniques. The [bag-of-words] technique, where each distinct word in a text is represented by one integer, is the simplest.

6. Tokenization - This simply refers to the procedure of turning a list of regular text strings into a collection of tokens.

7. Word Embedding - it is a feature vector representation of words which are used for other natural language processing applications. It can identify a word's position in a document, its semantic and syntactic similarities, its relationship to other words, etc. We will use GloVe Embedding.

8. We will then train the model using LSTM (Long short-term memory) which is a sequence model. If we look at the dataset, we see that some words are predominantly featured in both positive and negative tweets, this can be problematic for models such are Naïve bayes, SVD, etc…. That is why we will use a Sequence model, which requires sequential data.

9. We will then optimize the model by using an optimization algorithm for gradient descent and use call-backs to improve our results at the end of each epoch.

10. We will then evaluate our model by looking at the learning curve of loss and accuracy for the model on each epoch. We will also create a confusion matrix for an overlook at the models performance and finally look at the classification scores, we we will be able to see precision, recall, f1 score and support.

References

GeeksforGeeks, 2022. *Introduction to Recurrent Neural Network - GeeksforGeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/> [Accessed 12 September 2022].

Saeed, M., 2022. [online] Available at: <https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/> [Accessed 13 September 2022].

Biswaal, A., 2022. [online] Available at: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn> [Accessed 14 September 2022].

Dobilas, S., 2022. *RNN: Recurrent Neural Networks — How to Successfully Model Sequential Data in Python*. [online] Medium. Available at: <https://towardsdatascience.com/rnn-recurrent-neural-networks-how-to-successfully-model-sequential-data-in-python-5a0b9e494f92> [Accessed 14 September 2022].

Awan, A., 2022. [online] Available at: <https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network> [Accessed 15 September 2022].