# Long Short Term Memory Networks for NLP & Time Series Prediction

## 1. Introduction

Long Short Term Memory networks (LSTMs) are a way to work with sequence information. Often this information is in Language Processing and Time Series Data. The two examples we will be looking at are for each of these. The first is analyzing text to predict what type of article it is and the second is to predict the number of airline passengers in the future.

## 2. LSTM for Natural Language Processing (NLP)

### 2.1. Data preparation

The first step is to load in the stop words. These words will be taken out of the text to reduce redundant or unimportant words and information. This way, each word has a higher 'value' for the model. The next step was to split the articles into a training and testing data set. After that I tokenized all the words and kept the top 5000 most common words. Tokenizing items means to put a number to the word. Taking the top 5000 lets us use information that can be learned on. Some word that is mentioned once isn't useful and can't necessarily be learned on for the future. The tokenizing is done on the training data to prevent data leakage.

### 2.2. Modelling

The model uses a LSTM layer with a bidirectional shape. This way the data can be learned on from past - future, but also future - past. I then train the model using 'sparse_categorical_crossentropy' with an Adam optimizer. I did 10 epochs and got a validation accuracy of 94%.

## 3. LSTM for Time Series

### 3.1. Data preparation

The data is prepared in a similar way as the NLP example. The data here is scaled from 0-1. The way that this model works is by training on 'lag' variables. These variables represent n periods before it. It can train on just the previous observation or the previous n observations.

### 3.2. Modelling

In this LSTM model, we don't use bidirectional because we only care about past - future. We run the first model with 100 epochs and get a testing Root Mean Squared Error (RMSE) of 47.6. The next model goes from 1 -> 3 lag variables and gets a RMSE of 69.74. The next model uses time steps instead of subsequent observations. This keeps it consistent. The test RMSE is 63.3. The final model keeps LSTM memory remembered between batches and got a testing RMSE of 49.0. The final model used multiple layers of LSTM and got a testing RMSE of 108.9. In these models each of these features gets added on top of another. Let this be an example of how complexes != performance. More training epochs might help these issues. The model needs to 'fit' the data.

## 4.Conclusion

These two experiments offer a short introduction into using LSTMs to predict sequence data. More importantly, it is an example of how the data should be sequences and cleaned before using it in a model. The data cleaning/engineering was the most difficult part to understand.

### 4.1.Future tasks

4.1.1. Build this model into a class structure for future usability. Build The differentiations in this into the class and functions.