SENTIMENT ANALYSIS

What is Sentiment Analysis?

Sentiment analysis (or opinion mining) is a natural language processing technique used to interpret and classify emotions in subjective data. Sentiment analysis is often performed on textual data to detect sentiment in emails, survey responses, social media data, and beyond.

Why do we need Sentiment Analysis:

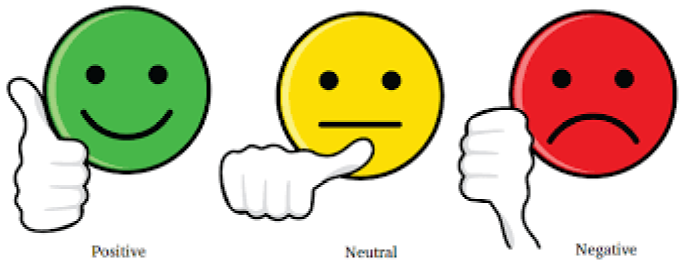
There are a lot of applications of sentiment analysis.

- Social media monitoring, we can get general response of people on a tweet or post on social media.
- Customer support, sentiment analysis helps to better analyse customers sentiments on product and analyse what changes are needed to be done in product.
- Sentiment Analysis helps in analysing the market trends.
- Sentiment analysis of movie reviews tells how are people reacting towards the movie, is there sentiment positive or negative.

Project Details:

In the project we will perform Sentiment Analysis using Recurrent Neural Networks that will help in analysing the sentiment of viewers from the movie review comments. We are using Recurrent Neural Networks instead of feedforward network because this provides more accuracy as we are able to keep track of sequence of words.

Dataset: We will be using a dataset of movie reviews in which we have labels positive or negative accompanied with it.

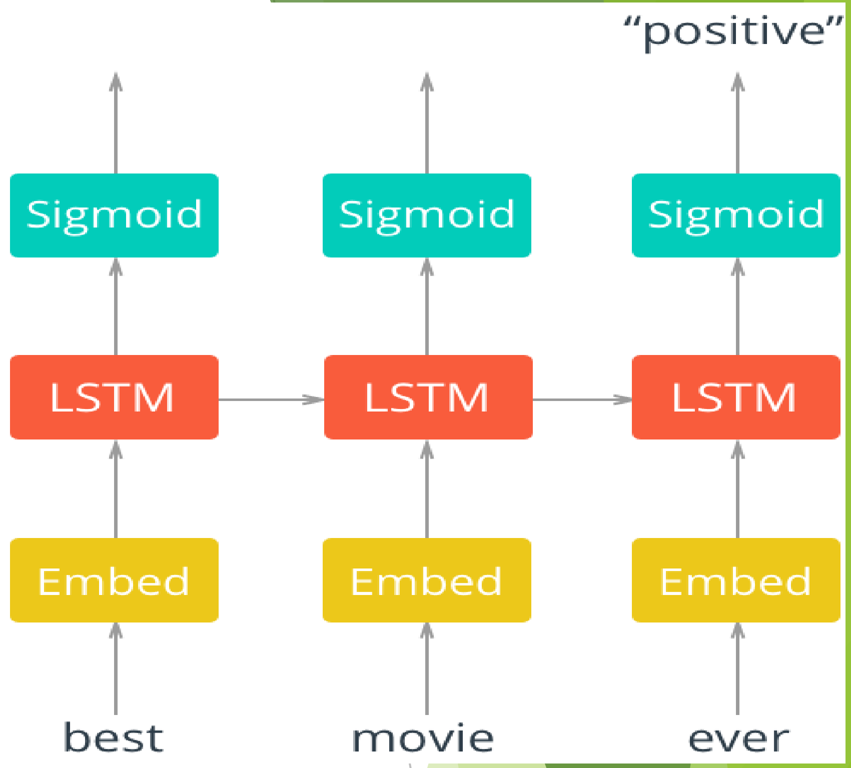


Network Architecture

- **First, we'll pass in words to an embedding layer.** We need an embedding layer because we have tens of thousands of words, so we'll need a more efficient representation for our input data than one-hot encoded vectors.

- **After input words are passed to an embedding layer, the new embeddings will be passed to LSTM cells.** The LSTM cells will add *recurrent* connections to the network and give us the ability to include information about the *sequence* of words in the movie review data.

- **Finally, the LSTM outputs will go to a sigmoid output layer.** We're using a sigmoid function because positive and negative = 1 and 0, respectively, and a sigmoid will output predicted, sentiment values between 0-1.



Steps to define the model-

- Firstly we will load the data.
- Then we will perform some pre-processing on the data i.e. we will remove punctuations and new line characters and then combine them in one long string.
- Now, we will perform embedding to our data. Since we cannot passwords to model so we will be converting the data to integers. We will also encode our labels 1 as positive and 0 as negative.
- Now, we want our data to be of the same length before passing to the network. So we will set a length and reviews with lengths less than that will be padded with 0 and sequences with lengths more will be truncated.
- We have our data ready so now we can split it into training(80%), validation(20%), and test data(20%). Training data is the data on which we perform training. Test data is stored to check our trained model. Validation data is stored to avoid overfitting.

- Now we will define our model architecture and instantiate the model i.e define how many layers and parameters we will be passing. The layers are:
  - An embedding layer that converts our word tokens (integers) into embeddings of a specific size.
  - A fully-connected output layer that maps the LSTM layer outputs to a desired output_size
  - A sigmoid activation layer which turns all outputs into a value 0-1; return only the last sigmoid output as the output of this network.
- Now we will set our hyperparameters for the model.
  - Vocab size: Size of our vocabulary or the range of values for our input, word tokens.
  - Output size: Size of our desired output; the number of class scores we want to output (pos/neg).
  - Embedding dim: Number of columns in the embedding lookup table; size of our embeddings.
  - Hidden dim: Number of units in the hidden layers of our LSTM cells. Usually larger is better performance wise. Common values are 128, 256, 512, etc.
  - N layers: Number of LSTM layers in the network.

- Then we will set the learning rate, define the loss as Binary Cross Entropy Loss and train the model for few epochs.
- Now, we can test our model on the test set. We can also experiment with hyperparameters for good accuracy.
- Now, our model is ready, we can store it and then perform test on different reviews manually.

Further improvements:

We can design some web app and deploy our model through that app so that the model can be tested directly through a GUI.

Conclusion:

The sentiment Analysis model gives us the

Thank You!