ALGORITHM

HAN

The hierarchical attention network model that this project was based on was proposed in 2016 and has two distinctive characteristics which set it apart from other learning models. First, it has a hierarchical structure that mirrors the hierarchical structure of documents. Second, it has two levels of attention mechanisms applied at the word and sentence level, enabling it to attend differentially to more and less important content when constructing the document representation. That being said, there are 4 main components to the HAN architecture which include the word sequence encoder, word-level attention layer, sentence encoder, and sentence-level attention layer. The model was constructed using Keras, due to Keras providing the ability to create custom layers, which were used for the word and sentence level attention layers. We begin constructing the model by creating a word encoder using the pre trained word vectors provided by Glove. The word encoder creates an embedding via an embedding matrix after which we use a bidirectional gated recurrent unit (GRU) to get annotations of words by summarizing information from both directions for words, and therefore incorporating the contextual information in the annotation. Next, we create the word attention layer, which computes sentence vectors by extracting words that are important to the meaning of the sentence and aggregating the representation of those informative words. Using the sentence vector, we again use a Bidirectional GRU to encode the sentences which we then pass to the sentence attention layer. At the sentence attention layer, we introduce a second attention mechanism and a sentence level context vector to measure the importance of the sentences. This yields a document vector that summarizes all the information of sentences in a document. Finally, the document vector is a high-level representation of the document as a whole and is used as a feature set for document classification. Ultimately, the HAN model was able to achieve a maximum accuracy of 90.2% and a validation accuracy of 87.3%

CNN

CNN is a class of deep, feed-forward artificial neural network initially developed for Image Processing. It mainly involves 2 major operations – Convolution and Pooling. Output from Convolution and Pooling Layers is given to Fully Connected Layer which is in principle the same as traditional Multilayer Perceptron. High success in Image Recognition led to CNN being used in NLP tasks.

Different Layers used in CNN:

- Input – Each sentence is represented by the important vocabulary words in that sentence. These words make up the input and are fed to the network in batches.
- Embedding Layer – This layer is responsible to represent each word in the sentence in the form of a 100-dimensional vector. This vector representation of words is taken from the pretrained GloVe 100-d vectors.
- Conv1D Layer – In this layer, a Conv1D operation is applied to a window. These convolution filters generate meaningful features which are then passed to the pooling layer.
- Max. Pooling Layer – In this layer pooling operation is used to combine the vectors resulting from different convolution windows into a single vector. This is done by taking the maximum value observed in resulting vector from the convolutions. This vector will

capture the most relevant features of the document. This vector is then passed to the fully connected layer.

- Flatten Layer – This is the layer where every input is connected to every output. The vector from the max pooling layer is taken and passed to the output layer for making predictions.
- Output Layer – This layer uses a single neuron with sigmoid activation to generate binary predictions. This layer uses standard adam optimizer with binary cross-entropy loss function.

RNN

Recurrent nets are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies. These algorithms take time and sequence into account, they have a temporal dimension. In the case of feedforward networks, input examples are fed to the network and transformed into an output; with supervised learning, the output would be a label, a name applied to the input. That is, they map raw data to categories, recognizing patterns that may signal, for example, that an input image should be labeled “cat” or “elephant.”

Different Layers used in RNN:

- Input – Each sentence is represented by the important vocabulary words in that sentence. These words make up the input and are fed to the network in batches.
- Embedding Layer – This layer is responsible to represent each word in the sentence in the form of a 100-dimensional vector. This vector representation of words is taken from the pretrained GloVe 100-d vectors.
- Bidirectional Layer – In this layer, two hidden layers are connected in opposite directions to the same output.
- Max. Pooling Layer – In this layer pooling operation is used to combine the vectors windows into a single vector. This is done by taking the maximum value observed in resulting vector. This vector will capture the most relevant features of the document. This vector is then passed to the fully connected layer.
- Dense Layer – This is the layer where every input is connected to every output. The vector from the max pooling layer is taken and passed to the dropout layer.
- Dropout Layer – This layer is used to prevent the model from overfitting. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on responsibility for the inputs.
- Output Layer – This layer uses a single neuron with sigmoid activation to generate binary predictions. This layer uses standard adam optimizer with binary cross-entropy loss function.