GloVe word embeddings

We will be using GloVe embeddings. GloVe stands for "Global Vectors for Word Representation". It's a somewhat popular embedding technique based on factorizing a matrix of word co-occurence statistics.

Specifically, we will use the 100-dimensional GloVe embeddings of 400k words computed on a 2014 dump of English Wikipedia.

## Approach

Here's how we will solve the classification problem:

* convert all text samples in the dataset into sequences of word indices. A "word index" would simply be an integer ID for the word. We will only consider the top 5,000 most commonly occuring words in the dataset, and we will truncate the sequences to a maximum length of 300 words.
* prepare an "embedding matrix" which will contain at index i the embedding vector for the word of index i in our word index.
* load this embedding matrix into a Keras Embedding layer, set to be frozen (its weights, the embedding vectors, will not be updated during training).
* build on top of it a LSTM neural network, ending in a softmax output over our 5 categories.

## Preparing the text data

(1)read txt – readlines .We will also set up a list of labels by add 20000 to each(since the input data is well prepared.)

(2) Then we can format our text samples and labels into tensors that can be fed into a neural network. To do this, we will rely on Keras utilities keras.preprocessing.text.Tokenizer and keras.preprocessing.sequence.pad\_sequences.Tokenizing with tokenizer function in Keras. preprocessing. Fit the txt data in to.

(3)pad to sequence

(4) Next, we compute an index mapping words to known embeddings, by parsing the data dump of pre-trained embeddings:Read the glove.txt and make it into a dictionary of words.

(5) At this point we can leverage our embedding\_index dictionary and our word\_index to compute our embedding matrix:Define embedding matrix.

We load this embedding matrix into an Embedding layer. Note that we set trainable=False to prevent the weights from being updated during training.

All that the Embedding layer does is to map the integer inputs to the vectors found at the corresponding index in the embedding matrix, i.e. the sequence [1, 2] would be converted to [embeddings[1], embeddings[2]]. This means that the output of the Embedding layer will be a 3D tensor of shape (samples, sequence\_length, embedding\_dim).

1. Build up a simple sequential model which contains the embedding layer and a layer of LSTM ready to train.
2. REFERRENCE:https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html