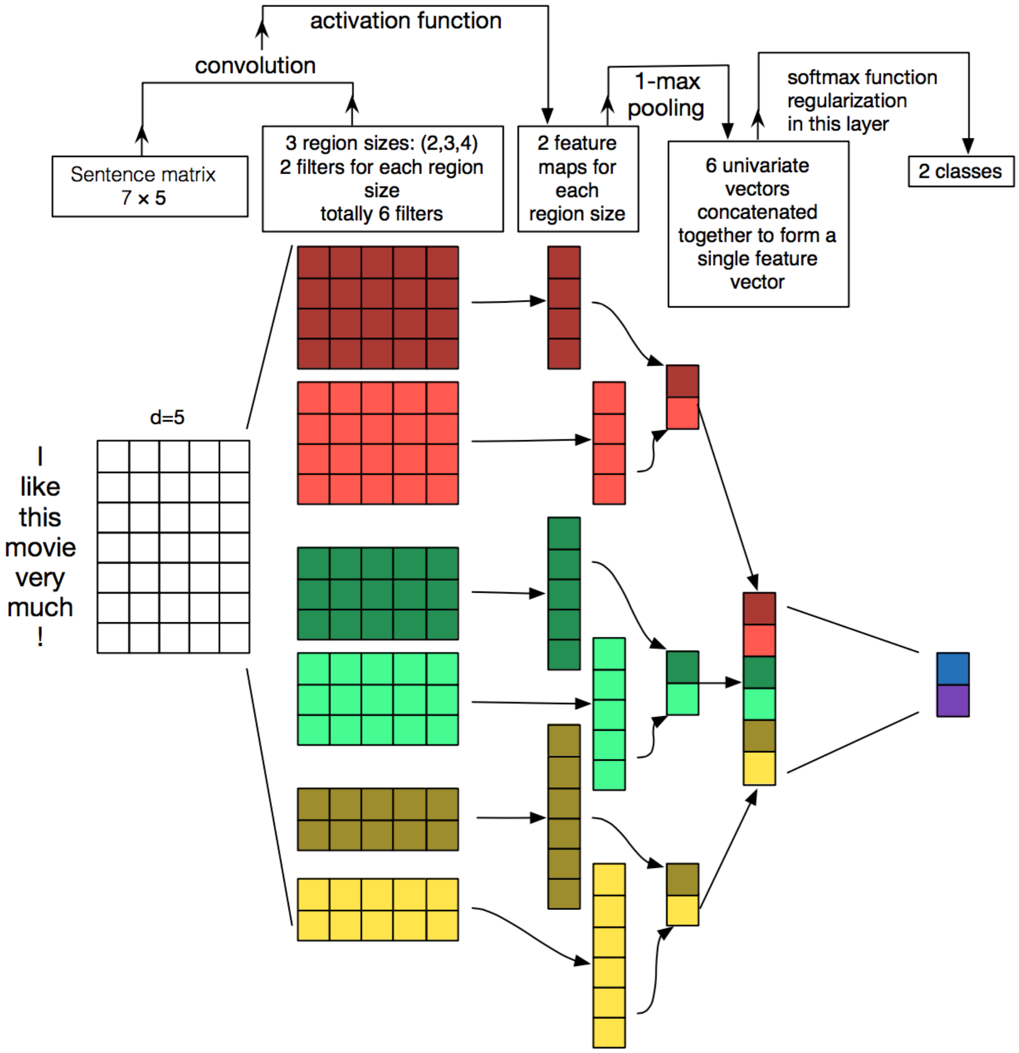
CNN are used for matrix data while RNNs are used for Sequence Data, so we need to first preprocess the data into a 2d matrix rather than the Sequence RNN uses

To convert a sentence to a two-dimensional tensor we can use each row as the token of a word.
Just an example

How are you?

Word	Token 1	Token 2	Token 3
How	1.2	2.3	1.3
are	1.3	4.3	9.8
you	1.6	1.0	2.2

And we can use the model class used for RNN to implement the CNN class.


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

The class CNN has a constructor, that initializes all the values

1. A parser function read the input and the creates tokens for the word in the input, and pads sentences depending upon the max sentence length, and use the vocab from the dataset class
2. A build function that builds a neural network using TensorFlow with an input shape depending on the size of the token,
   The first layers embeds words into low-dimensional vectors. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time. Next, we max-pool the result of the convolutional layer into a long feature vector, add dropout regularization, and classify the result using a softmax layer

   a.
   ```
   model = models.Sequential()
   model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(64, (3, 3), activation='relu'))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(64, (3, 3), activation='relu'))
   ```

   and adding layers to the CNN such as MaxPooling and Conv2D with different filter size because each convolutional layer produces a tensor of different size

   b. Then we add a drop out layer.

   c. Using the feature vector from max-pooling (with dropout applied) we can generate predictions by doing a matrix multiplication and picking the class with the highest score. We could also apply a softmax function to convert raw scores into normalized probabilities, but that wouldn’t change our final predictions.

   d. Using our scores we can define the loss function. The loss is a measurement of the error our network makes, and our goal is to minimize it. The standard loss function for categorization problems it the cross-entropy loss.