Duration of Project - 22 May to 7 July

Presented by: Pooja Dashottar

pooja95teen@gmail.com

IIEST, Shibpur

Text processing using CNN

Sentiment analysis of sentences using ML to classify into diffeterent categories

# Acknowledgement

*I am Pooja Dashottar, a student of Computer Science and Technology in IIEST, Shibpur. The internship opportunity I had with SkyBits Technology Pvt. Ltd. , Bangalore was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it.*

*I am using this opportunity to express my deepest gratitude and special thanks to my mentor in SkyBits, Mr. Tirthankar Banerjee who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path. His careful and precious guidance were extremely valuable for my study both theoretically and practically. He also helped me in doing a lot of Research and I came to know about so many new things, I am really thankful to him.*

*I express my deepest thanks to the Director of IIEST, Shibpur, Prof. Ajay Kr. Roy for giving me the golden opportunity to do this wonderful project on the topic* “Text Processing in CNN”.

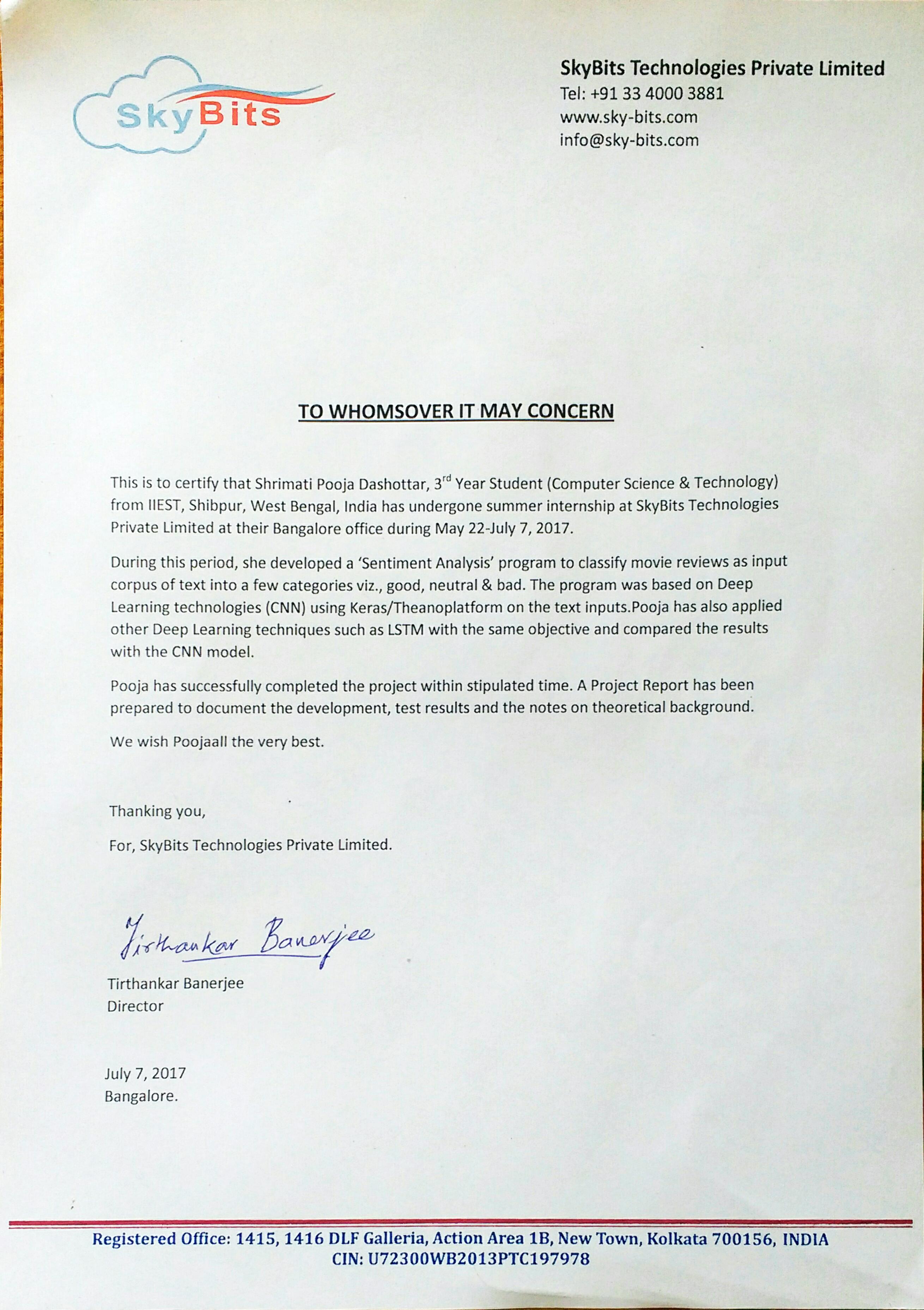
*This opportunity has set a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement to attain desired career objectives. Hope to continue cooperation with all of you in the future.*

*Sincerely,*

*Pooja Dashottar*

*Bangalore*

28/06/2017



Contents

[Acknowledgement 1](#_Toc487279838)

[Objective 4](#_Toc487279839)

[Approach 4](#_Toc487279840)

[Software and Tools 4](#_Toc487279841)

[Virtual Environment 5](#_Toc487279842)

[Installation 5](#_Toc487279843)

[Technical Description 6](#_Toc487279844)

[Text Preprocessing: 7](#_Toc487279845)

[Compilation, Training and Evaluation 7](#_Toc487279846)

[Model Saving 8](#_Toc487279847)

[Model Testing 8](#_Toc487279848)

[Results 8](#_Toc487279849)

[Long Short-Term Memory (LSTM) 10](#_Toc487279850)

[What’s Next ? 11](#_Toc487279851)

[Annexure 12](#_Toc487279852)

[Natural Language Processing (NLP) 12](#_Toc487279853)

[Major Evaluations and Tasks 12](#_Toc487279854)

[Spacy 12](#_Toc487279855)

[Introduction to convolutional neural network 12](#_Toc487279856)

[Hyperparameters 13](#_Toc487279857)

[The pooling step 15](#_Toc487279858)

[Fully Connected Layer 16](#_Toc487279859)

[Multi Layer Perceptron 16](#_Toc487279860)

[Training our MLP 17](#_Toc487279861)

[BackProp Algorithm 18](#_Toc487279862)

[How does CNN apply to NLP ? 18](#_Toc487279863)

[Dropout Regularization Technique 19](#_Toc487279864)

[Long Short-Term Memory (LSTM) 20](#_Toc487279865)

[The core idea behind LSTMs 20](#_Toc487279866)

[References 21](#_Toc487279867)

# Objective

Using Natural Language Processing and Convolutional Neural Network, machine learning method to extract, identify and characterize the sentiment content of a text unit.

* We have performed a binary sentiment classification by creating a learning model which implements Convolutional Neural Network (ConvNet / CNN)
* Came up with best values of hyperparameters to get best accuracy of the model.

# Approach

We need to implement a CNN model which can be trained by a labeled dataset and then can be used to classify unlabeled dataset with high accuracy. We have performed our task step by step as given below :

* For training the model a labeled dataset is taken. This dataset will be fed as input to the CNN model.
* Text Preprocessing
  + Unnecessary punctuations and case sensitive is cleared from the dataset.
* Word Embedding
  + A CNN model can only take numbers as input to the model therefore the words present in the dataset needs to be converted into vectors keeping the syntactical information of the words as it is.
* Creating CNN model
  + Creating a CNN model includes number of steps which are mentioned here –
  + Convolution step
  + ReLU / Non-linearity
  + Max Pooling / Sub sampling
  + Classification / Fully Connected layers
* To prevent the model from overfitting while training, Dropout Regularization Technique is used.
* Model compilation
* Model training using the train data (all data are in the form of vectors).
* Model Validation
* Model Saving
* Model Testing
* Checking the results by changing hyperparameters of CNN.
* Coming up with best values of hyperparameters to get best accuracy.

# Software and Tools

* Anaconda python 2.7 package in Windows 10.
  + <https://www.continuum.io/downloads>
* IDE used is Jupyter Notebook which is inbuilt with the anaconda package.
* Virtual Environment
* Spacy API
* Keras over Theano

## Virtual Environment

All the installation is done on a virtual environment. Therefore, before going to installation part, we need to create a virtual environment for our work.

**Virtual Environment** is a tool to create isolated Python environments. The basic problem being addressed is one of dependencies and versions, and indirectly permissions. Virtual environment helps to keep modules of different projects separately and to put those projects into the cloud.

We will do all our work in a virtual environment.

How to create a virtual environment in anaconda ?

In Anaconda prompt –

$ conda create -n <envname> anaconda

It’ll ask to install all the packages which were installed during default anaconda installation.

To check all the env. Present, in anaconda prompt type –

*$ conda info -e*

Every time while using any virtual environment its necessary to first activate the environment.

*$ activate <envname>*

To deactivate all the virtual environments,

*$ deactivate*

To lists all the installed packages -

*$ conda list*

To remove an environment -

*$ conda remove -n <envname> --all*

## Installation

All installations are done after activating the virtual environment.

* Spacy Installation

For proper installation, use superuser mode.

In Anaconda Prompt -

*$ runas /user:administrator “<command>”*

Commands :

*$ pip install -U spacy* or

*$ conda install -c conda-forge spacy=1.8.2*

Also, install Microsoft Visual C++ redistributable package which is used by Spacy

https://www.microsoft.com/en-us/download/confirmation.aspx?id=2092

Spacy has different modules , We will work with English module.

$ python -m spacy download en

* Installing Keras over Theano –
  + Theano

*$ conda install mingw libpython*

*$ pip install theano*

To upgrade to the newer version of theano :

*$ conda install git*

*$ pip install –upgrade –no-deps git+https://github.com/Theano/Theano.git*

* + Keras

*$ pip install git+https://github.com/fchollet/keras.git*

To check all installed packages-

*$ conda list*

It shows the versions as well –

* Theano – 0.9.0
* Keras – 2.0.4
* Spacy – 1.8.2

By default keras runs under Tensorflow. To make it run under Theano open the file keras.json by typing,

*$ %USERPROFILE% /.keras.keras.json*

Change the backend to Theano as default and save it.

* The entire code is present in the link given below :

https://github.com/pdashottar/Text-Processing

# Technical Description

We are working in Jupyter IDE. Jupyter is a web-based interactive development environment. It supports multiple languages like Octave, Python, R etc.

Labeled IMDB dataset is available online. It has 2 labels (1,0), where 1 representing a positive review and 0 a negative review.

A suitable data needs to be fed as input to our CNN model. CNN model can take numbers as input, so the words have to be converted into numbers appropriately. Therefore, text preprocessing is required.

## Text Preprocessing:

Not all data in the form given as review is relevant for our task. Only the words are important. So, the data need to be cleaned and then it should be converted into vectors.

Cleaning involves making the data case insensitive and removing all punctuations.

Word embedding : It is a method of extracting features out of text so that we can input those features into our ML model to work with text data in ML. It tries to save the syntactical information of words. Every word has a vector.

Using Spacy modules, words can be vectorized into 300 Dimension (standard)

After getting the vector form of words, we are all set to define our CNN model.

## Compilation, Training and Evaluation

Before training a model, we need to configure the learning process, which is done via the **compile** method.

After compilation model can be plotted in a PNG file.

In ML data is divided into 3 parts –

1. Training set
2. Validation set - to estimate how well your model has been trained
3. Testing set

Epoch is the no. of iterations performed for training the data. As the no. of iteration increases, the accuracy of our classification also increases. Epoch is directly proportional to accuracy.

In the neural network terminology:

* one **epoch** = one forward pass and one backward pass of *all* the training examples
* **batch size** = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
* number of **iterations** = number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).

Example: if we have 1000 training examples, and our batch size is 500, then it will take 2 iterations to complete 1 epoch

* Loss = The lower the **Loss**, the better a model (unless the model has over-fitted to the training data). The loss is calculated on **training** and **validation** and its interpretation is how well the model is doing for these two sets. Loss is not in percentage as opposed to accuracy and it is a summation of the errors made for each example in training or validation sets.

After training, model evaluation is performed on test dataset to rectify the errors occurred in testing.

## Model Saving

After training and evaluation, the model needs to be saved so that it can be used again and again. A model is saved in HDF format. Hierarchical Data Format (HDF) is a set of **file** formats (HDF4, **HDF5**) designed to store and organize large amounts of data. After saving, the model can be loaded from some other program as well and can be used for testing.

## Model Testing

Model is tested in two ways –

1. Separating out 20% of dataset for testing purpose and giving each sentence as the input to the model and then comparing the predicted class with the actual class of all sentences. It can easily calculate the accuracy of unknown data.
2. Giving sentence as user input to the input and then checking out the prediction. Similar kind of words can be used to test the prediction.

Using Spacy synonyms of a word can be found ,

*def most\_similar(word):*

*by\_similarity = sorted(word.vocab, key=lambda w: word.similarity(w), reverse=True)*

*return [w.orth\_ for w in by\_similarity[:10]]*

For our testing purpose similar words for ‘good’ results in –

*print(most\_similar(nlp.vocab[u'good']))*

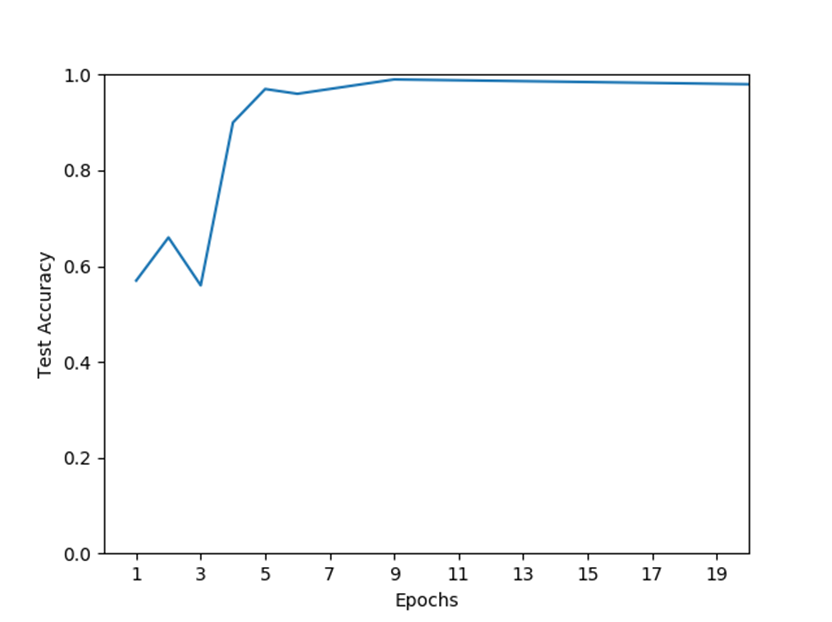
Results in -

*[u'good', u'Good', u'GOOD', u'great', u'Great', u'GREAT', u'better', u'Better', u'BETTER', u'very']*

## Results

The accuracy of testing data is our main concern. We have changed some hyperparameters to the CNN model and checked the accuracy. We have come up with a few conclusions –

1. On increasing the number of epochs the accuracy of the training data increases. But with large no. of epochs (>20) the model starts overfitting.



1. On increasing the validation split , the accuracy increase.
2. On increasing the batch size, the accuracy increase.

We have come up with optimum hyperparameters values to get good accuracy on test dataset for our IMDB dataset.

filter\_size (2,3,4,5,6)

pool dropout 0.8

dense dropout 0.8

stride 1

batch size 64

num filters 5

epochs 5

Still the model cannot understand ‘good’ and ‘not bad’ as same.

By checking the similarity between the two vectors corresponding to ‘good’ and ‘not bad’, the similarity value comes out to be very small.

Similarity between two words/phrases can be found out by a function available in Spacy.

For obj1 & obj2

obj1.similarity(obj2)

Also, the cosine distance between vectors corresponding to ‘good’ and ‘not bad’ can be calculated. We have found that the distance is comparable to that between ‘good and ‘bad’, Hence the model fails to understand that ‘good’ and ‘not bad’ are same. Also, since vector corresponding to ‘not bad’ and ‘bad not’ are same therefore the model cannot understand the context of a word. For doing so, word2vec can be used, which is not discussed in this project.

# Long Short-Term Memory (LSTM)

Since LSTM is good for remembering the context of a word. We have tried to perform sentiment analysis using LSTM model and check the accuracy.

Hyperparameter values used –

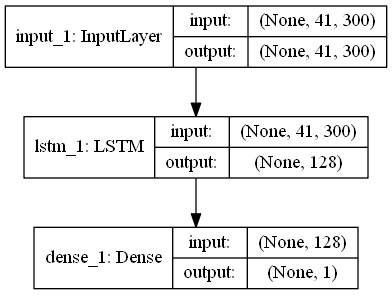
Batch size=32

Test size= 0.25 of train size

dropout=0.5, recurrent dropout=0.5

epochs=4

After compilation, the model can be plotted -



I have used 1 LSTM and 1 Dense layer however multiple layers can be put one after other to get better results. We have got good results with the model given above.

Results –

Train accuracy: 0.9360

Validation accuracy: 0.9687

Test accuracy: 0.968666666667

# What’s Next ?

* The CNN model cannot learn the context of words depending upon the sentence. Recursive neural networks can learn about the context from previous words in a sentence. Hence it can predict better. Therefore, RNN can be used instead.
* For converting words into vectors, we have used the standard 300D space in which each vector corresponds to a word from the entire English dictionary, therefore there’s not much of vector distance between words having opposite semantics. If the vector space would be chosen in such a way that only the words present in the dataset would comprise the entire vector space then the vector distances between words having opposite semantics would be much higher hence resulting in more accurate testing results. Creating our own vector space is possible in spacy.

# Annexure

## Natural Language Processing (NLP)

**Natural language processing (NLP**) is a field of [computer science](https://en.wikipedia.org/wiki/Computer_science), [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) and [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) concerned with the interactions between [computers](https://en.wikipedia.org/wiki/Computer) and [human (natural) languages](https://en.wikipedia.org/wiki/Natural_language), and, in particular, concerned with programming computers to fruitfully process large [natural language corpora](https://en.wikipedia.org/wiki/Corpus_linguistics). Challenges in natural language processing frequently involve [natural language understanding](https://en.wikipedia.org/wiki/Natural_language_understanding), [natural language generation](https://en.wikipedia.org/wiki/Natural_language_generation) (frequently from [formal, machine-readable logical forms](https://en.wikipedia.org/wiki/Formal_language)), [connecting language and machine perception](https://en.wikipedia.org/wiki/Symbol_grounding_problem), [managing human-computer dialog systems](https://en.wikipedia.org/wiki/Dialog_system), or some combination thereof.

### Major Evaluations and Tasks

* Part-of-speech tagging
* Tokenization
* Named entity recognition
* Parsing

### Spacy

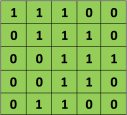
Spacy is a python API for Natural language processing in Python. One great feature which is useful for us is that Spacy API offers named entity recognition and ready access to word vectors. Since input to CNN model needs to be numbers therefore our text needs to be converted into vectors.

## Introduction to convolutional neural network

What is Convolution ?

ConvNets derive their name from the [“convolution” operator](http://en.wikipedia.org/wiki/Convolution). Convolution is a sliding window function applied to a matrix. The primary purpose of Convolution in case of a ConvNet is to extract features from the input data. Convolution preserves the spatial relationship between pixels by learning data features using small squares of input data.

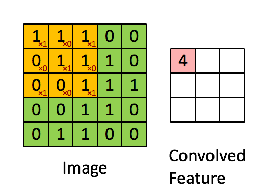
 Every input data can be considered as a matrix of pixel values. Consider a 5 x 5 matrix whose pixel values are only 0 and 1



Also, consider another 3 x 3 matrix as shown below:



Then, the Convolution of the 5 x 5 matrix and the 3 x 3 matrix can be computed as shown below:



We slide the orange matrix over our original image (green) by 1 pixel (also called ‘stride’) and for every position, we compute element wise multiplication (between the two matrices) and add the multiplication outputs to get the final integer which forms a single element of the output matrix (pink). Note that the 3×3 matrix “sees” only a part of the input image in each stride.

In CNN terminology, the 3×3 matrix is called a ‘**filter**‘ or ‘kernel’ or ‘feature detector’ and the matrix formed by sliding the filter over the image and computing the dot product is called the ‘Convolved Feature’ or ‘Activation Map’ or the ‘**Feature Map**‘. It is important to note that filters act as feature detectors from the original input matrix.

It is evident from the above example that different values of the filter matrix will produce different Feature Maps for the same input matrix.

In practice, a CNN learns the values of these filters on its own during the training process (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process). The more number of filters we have, the more features get extracted and the better our network becomes at recognizing patterns in unseen data.

### Hyperparameters

Some common CNN hyperparameters which are changed to obtain an optimum model accuracy are –

* Filter Size
* Epochs
* Batch size
* Number of Filters
* Strides
* Dropout

We will go through all these hyperparameters in the coming sections.

Size of the filters play an important role in finding the key features. A larger size kernel can overlook at the features and could skip the essential details in the images whereas a smaller size kernel could provide more information leading to more confusion. Therefore there is a need to determine the most suitable size of the kernel/filter.

The size of the Feature Map (Convolved Feature) is controlled by these parameters that we need to decide before the convolution step is performed:

**Depth**: Depth corresponds to the number of filters we use for the convolution operation. If we are performing convolution using three distinct filters, thus producing three different feature maps. We can think of these three feature maps as stacked 2d matrices, so, the ‘depth’ of the feature map would be three.

**Stride:**Stride isthe number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

CNN are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

CNN is a supervised learning technique and requires a large no. of dataset for training.

It consists of multiple copies of the same neuron which is roughly analogous to the abstraction of function in computer science. A convolutional neural network can learn a neuron once and use it in many places, making it easier to learn the model and reduce errors.

One very nice property of convolutional layers is that they’re composable. You can feed the output of one convolutional layer into another. With each layer, the network can detect higher-level, more abstract features.

A hidden layer [neuron](http://standoutpublishing.com/g/neuron.html) is a neuron whose output is connected to the inputs of other neurons and is therefore not visible as a network output (hence the term hidden layer).

• Number of input and output units is determined by dimensionality of data set

• Number of hidden units M is a free parameter

• Adjusted to get best predictive performance

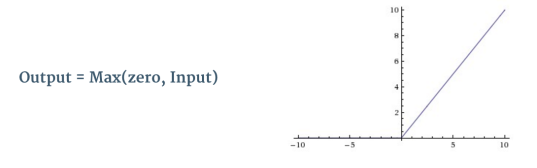
• Possible approach is to get maximum likelihood estimate of M for balance between under-fitting and over-fitting

There are four main operations in the ConvNet :

1. Convolution
2. Non Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

Convolution is already discussed earlier.

An additional operation called ReLU has been used after every Convolution operation. ReLU stands for Rectified Linear Unit and is a non-linear operation.



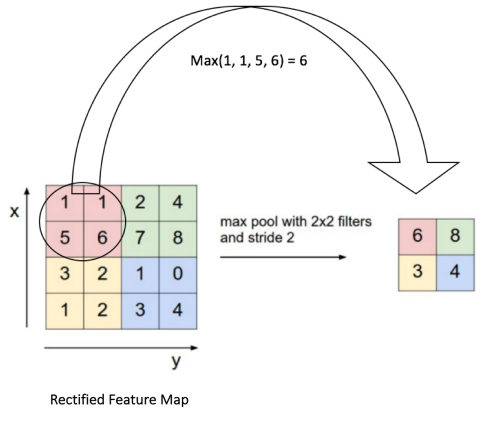
ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

### The pooling step

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

**Figure** shows an example of Max Pooling operation on a Rectified Feature map (obtained after convolution + ReLU operation) by using a 2×2 window.



We slide our 2 x 2 window by 2 cells (also called ‘stride’) and take the maximum value in each region. As shown in **Figure**, this reduces the dimensionality of our feature map.

Pooling operation is applied separately to each feature map. Due to this, we get three output maps from three input maps.

The function of Pooling is to progressively reduce the spatial size of the input representation. In particular, pooling

* makes the input representations (feature dimension) smaller and more manageable
* reduces the number of parameters and computations in the network, therefore, controlling [overfitting](https://en.wikipedia.org/wiki/Overfitting)
* makes the network invariant to small transformations, distortions and translations in the input data (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighborhood).
* helps us arrive at an almost scale invariant representation of our data.

Convolutional layer , ReLU and Pooling , each layer is repeated several times.

The output of the Pooling Layer acts as an input to the Fully Connected Layer, which we will discuss in the next section.

### Fully Connected Layer

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer.

The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer.

### Multi Layer Perceptron

A Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer).  While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non – linear functions.

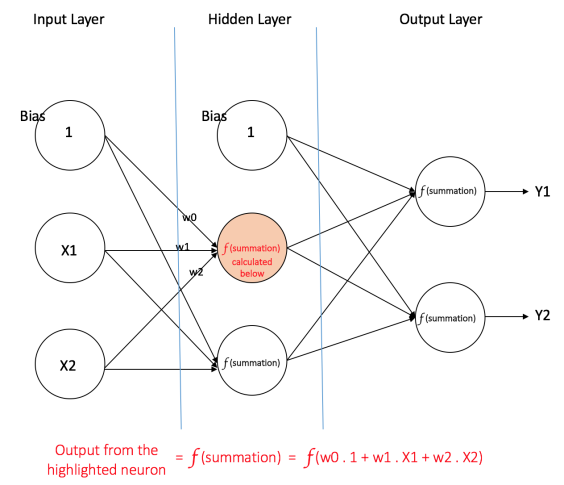
Figure shows a multi layer perceptron with a single hidden layer. Note that all connections have weights associated with them, but only three weights (w0, w1, w2) are shown in the figure.

**Input Layer:**The Input layer has three nodes. The Bias node has a value of 1. The other two nodes take X1 and X2 as external inputs (which are numerical values depending upon the input dataset). As discussed above, no computation is performed in the Input layer, so the outputs from nodes in the Input layer are 1, X1 and X2 respectively, which are fed into the Hidden Layer.

**Hidden Layer:**The Hidden layer also has three nodes with the Bias node having an output of 1. The output of the other two nodes in the Hidden layer depends on the outputs from the Input layer (1, X1, X2) as well as the weights associated with the connections (edges). Figure shows the output calculation for one of the hidden nodes (highlighted). Similarly, the output from other hidden node can be calculated. ***f***refers to the activation function. These outputs are then fed to the nodes in the Output layer.

**Output Layer:**The Output layer has two nodes which take inputs from the Hidden layer and perform similar computations as shown for the highlighted hidden node. The values calculated (Y1 and Y2) as a result of these computations act as outputs of the Multi Layer Perceptron.

Given a set of features **X = (x1, x2, …)**and a target **y**, a Multi Layer Perceptron can learn the relationship between the features and the target, for classification.



The output from the convolutional and pooling layers represent high-level features of the input data. The purpose of the Fully Connected layer is to use these features for classifying the input data into various classes based on the training dataset. We will see below how a multi layer perceptron learns such relationships.

### Training our MLP

The process by which a Multi Layer Perceptron learns is called the Backpropagation algorithm. BackProp is like "learning from mistakes". The supervisor corrects the ANN whenever it makes mistakes.   
  
An ANN consists of nodes in different layers; input layer, intermediate hidden layer(s) and the output layer. The connections between nodes of adjacent layers have "weights" associated with them. The goal of learning is to assign correct weights for these edges. Given an input vector, these weights determine what the output vector is. In supervised learning, the training set is labeled. This means, for some given inputs, we know(label) the desired/expected output.  

BackProp Algorithm

Initially all the edge weights are randomly assigned. For every input in the training dataset, the ANN is activated and its output is observed. This output is compared with the desired output that we already know, and the error is "propagated" back to the previous layer. This error is noted and the weights are "adjusted" accordingly. This process is repeated until the output error is below a predetermined threshold.   
  
Once the above algorithm terminates, we have a "learned" ANN which, we consider is ready to work with "new" inputs. This ANN is said to have learned from several examples (labeled data) and from its mistakes (error propagation).

In classification tasks, we generally use a **[Softmax function](http://cs231n.github.io/linear-classify/" \l "softmax" \t "_blank)** as the Activation Function in the Output layer of the Multi Layer Perceptron to ensure that the outputs are probabilities and they add up to 1. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

The last stage of a convolutional neural network (CNN) is a classifier.

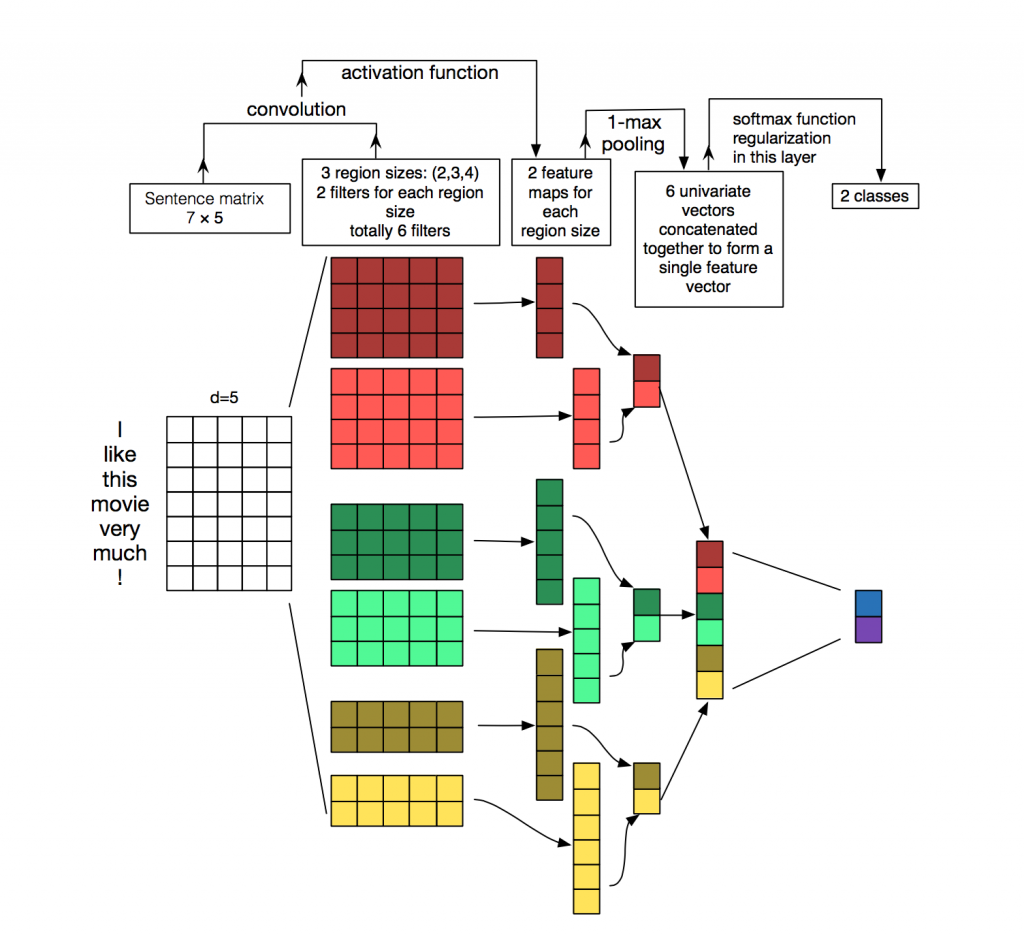
Our classifier needs individual features, just like any other classifier. This means it needs a feature vector.

Therefore, we need to convert the output of the convolutional part of the CNN into a 1D feature vector, to be used by the classifier part of it. This operation is called flattening. It gets the output of the convolutional layers, flattens all its structure to create a single long feature vector to be used by the dense layer for the final classification.

## How does CNN apply to NLP ?

the input to NLP tasks are sentences represented as a matrix. Each row of the matrix corresponds to one token, typically a word. That is, each row is vector that represents a word. Typically, these vectors are word embeddings. For a 10 word sentence using a 300-dimensional embedding we would have a 10×300 matrix as our input.

In NLP we typically use filters that slide over full rows of the matrix (words). Thus, the “width” of our filters is usually the same as the width of the input matrix. The height, or region size, may vary, but sliding windows over 2-5 words at a time is typical. Putting all the above together, a Convolutional Neural Network for NLP may look like this -



*Here we depict three filter region sizes: 2, 3 and 4, each of which has 2 filters. Every filter performs convolution on the sentence matrix and generates (variable-length) feature maps. Then 1-max pooling is performed over each map, i.e., the largest number from each feature map is recorded. Thus a univariate feature vector is generated from all six maps, and these 6 features are concatenated to form a feature vector for the penultimate layer. The final softmax layer then receives this feature vector as input and uses it to classify the sentence, here we assume binary classification and hence depict two possible output states.*

### Dropout Regularization Technique

**Regularization** is a *technique*used to solve the **overfitting**problem in statistical models.

Dropout is a regularization technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly to ensure that model doesn’t overfit. This works by preventing neurons from co-adapting and forcing them to learn individually useful features. It helps to view dropout as a form of ensemble learning. In ensemble learning we take a number of ‘weaker’ classifiers, train them separately and then at test time we use them by averaging the responses of all ensemble members. Since each classifier has been trained separately, it has learned different ‘aspects’ of the data and their mistakes are different. Combining them helps to produce a stronger classifier, which is less prone to overfitting. One ensemble variant is bagging, in which each member of the ensemble is trained with a different subsample of the input data, and thus has learned only a subset of the whole possible input feature space. Dropout can be seen as an extreme version of bagging. At each training step in a mini-batch, the dropout procedure creates a different network (by randomly removing some units), which is trained using backpropagation as usual.

Dropout is only used during the training of a model and is not used when evaluating the skill of the model. A dropout probability too low has minimal effect and a value too high results in under-learning by the network.

This is necessary for our model since this data set is a bit small so we’re likely to overfit with a powerful model.

The dimension of output layer = no. of labels. We have 2 labels 0 & 1.

## Long Short-Term Memory (LSTM)

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



### The core idea behind LSTMs

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.



The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

An LSTM has three of these gates, to protect and control the cell state.

# References

These are the few references which I have used for understanding Spacy and CNNs –

<http://textminingonline.com/getting-started-with-spacy>

<https://colah.github.io/posts/2014-07-Understanding-Convolutions/>

<http://deeplearning.net/tutorial/lenet.html>

<https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

<https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/>

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[www.iiests.ac.in/](http://www.iiests.ac.in/)

www.sky-bits.com/