Effective Use of Word Order for Text Categorization using Convolutional Neural Networks

Contents

1. Introduction

Text categorization is nothing but categorizing a large corpus of documents into predefined categories depending on the nature of the content. This categorization may be topic classification (to detect multiple topics like fiction, comedy, etc), spam email classification, sentiment classification (based on the polarization of the content, classification will be done in it to positive, negative, and neutral categories for a product or movie review), etc.

Text categorization can be utilized for solving many real-time natural language processing problems. For text categorization traditional and old approach involves representing words using bag of words then it would be given as input to a classification model like SVM but the problem with bag of words representation is it cannot consider word order. It just tries to consider the attendance of the word for a given sentence.

To overcome this issue convolutional neural networks (CNN) are implemented in this project to benefit from word order. In deep learning, CNN is a class of artificial intelligence often applied to solving image classification problems.

CNN makes use of the internal structure of the image through convolution in which each computation corresponds to a small region of the whole image. In this project, CNN was applied to text categorization to utilize the 1 d form text data so that individual units in CNN process the small region (Sequence of words) of the large document.

In this project following 2 types of Convolutional neural networks are implemented to effectively consider word order for text categorization.

1. Sequential Convolutional Neural Networks.
2. Extension Parallel Convolutional Neural Networks.

2. Bag of words Approach for Text Classification:

In general, any machine learning model does not understand words or sentences, so these texts and sentences need to be converted into some numerical or vector format that the computer can understand. The most used approach for this conversion is utilizing bag of words. In bag of words representation, it makes use of the dictionary which is nothing but a set of unique words. Based on this set of unique it tries to represent a sentence based on the attendance of words within the sentence for a given dictionary.

For example, consider the dictionary with size 5 with words

{“I am”, “Good”, “Person”, “hard”,” working”}

The representation of sentences

“I am Good” and

“I am hard working person “Would be as follows

Sentence	I am	Good	Person	hard	Working
I am Good	1	1	0	0	0
I am hard working person	1	0	1	1	1

Table 1(a)

So, in this way bag of words converts its sentences using unique words in the dictionary.

2.1 Problems with Bag of words representation:

In bag of words representation (Table 1 (a)) only considers the attendance of the words concerning the given dictionary but it does not consider word order which is very important in Text Categorisation.

For example, let us consider following movie reviews for sentiment analysis.

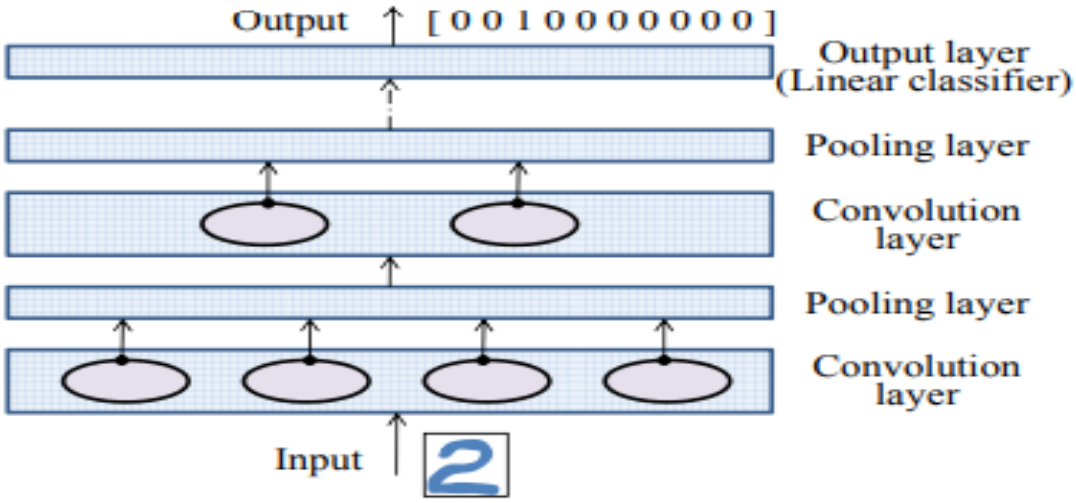
Review 1: “This is a good movie but I am not interested in movies”.

Review 2: “This is not a good movie but I am interested in movies.”

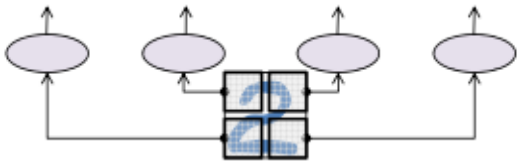
Review 1 is a positive review and Review 2 is a negative review based on the meaning. For both reviews' bag of words, representation would be the same because it does not consider word order. The meaning of the review changes with the placement of not in both reviews. In a bag of words, just the frequency of the “not” word is considered but not its placement. So, when both reviews have the same representation, if it is fed into a classification model then both reviews would be classified into the same sentiment label. An alternative to overcome this issue is by using higher order n-grams for classification but often dictionary size would be huge in NLP tasks it will not be feasible by representing groups of 2 to 3 words or higher order n-grams which makes the entire process complicated.

3. Traditional CNN for Images

CNN are feed-forward neural networks that involve several convolutions and pooling layers until the top layer in which it performs classification.



In CNN it has multiple processing units each of which processes small parts as region vectors for the large image, usually, a filter of a predefined size will be moved across with sliding length (stride length) to compute the small parts of the image with a non-linear function. These region vectors represent the small parts of the image usually involves in the concatenation of pixels in that region. For example, if the region size is 10*10 with 2 channels (white and black) then



it would be 200 dimensional.

One of the key features of convolutional neural networks is its capability to share a weight matrix within the layer while processing the input. For a given input X and location L which is a part of X, it would be computed as follows

$$\sigma (W \cdot r`(x) + b)$$

where r`(x) is the region vector of location L of X.

σ is a predefined component-wise non-linear function.

W is the matrix of weights

b is bias.

Weights and biases for a given layer will be learned during the training process. The main advantage of this weight sharing is it tries to identify the object in an image irrespective of its location in the image. For example, if an image has the ball at the top right corner, then it tries to learn the pattern of the ball, and when an image with the ball which is at the bottom right corner CNN can be able to understand the pattern of the ball because of the shared weights.

After going through convolutions, the output produced will be passed to the pooling layer which it tries to shrink the data by merging neighborhood pixels. These pooling operations can be

1. Max pooling: selects the max value for the given pooling size.

2. Global Max pooling: Selects the maximum out of the entire output filtered image.

3. Averaging: This takes the average value for the given pooling size.

3. CNN for Text

Above CNN for images can be applied to text data. Let us consider a document D= {w1, w2,w3….} with vocabulary V. To process the text data into the numerical form we need some form of vector representation of text data that preserves the internal locations of the document. One strategy would be to assume each word as a pixel which forms an image with D*1 pixels with V dimensions where V is the vocabulary. Each pixel can be represented as V dimensional

one hot encoded vector. This can be understood better with an example.

Let us consider Vocabulary V= {“don’t”,” hate”,” I” ,”it” ,”Love”} and Document as D= “I love it” this can be represented as below one hot encoded vector.

x = [0 0 1 0 0 | 0 0 0 0 1 | 0 0 0 1 0]

4. Sequential CNN

CNN for the image has regions as a concatenation of pixels, which creates p * v dimensional region vectors where p is the region size. Similarly, we can apply the same for text data let us consider above-mentioned document x with region size (which is predefined) as 2 with stride size 1 it would result in two regions like “I Love” and “Love it”. Vector representation for these 2 regions in one hot encoded format would be as follows

$$\mathbf{r}_0(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ — \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \end{bmatrix} \begin{matrix} \text{don't} \\ \text{hate} \\ \mathbf{I} \\ \text{it} \\ \text{love} \\ \\ \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \mathbf{love} \end{matrix} \qquad \mathbf{r}_1(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ — \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \end{bmatrix} \begin{matrix} \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \mathbf{love} \\ \\ \text{don't} \\ \text{hate} \\ \text{I} \\ \mathbf{it} \\ \text{love} \end{matrix}$$

After the creation of region vectors then the remaining process would be the same as traditional CNN. Feature vectors are generated

from text regions then after performing convolutions it tries to convert embedded text regions to low dimensional vector space.

A neural network that involves the above-mentioned input data representation for text data would be considered as sequential CNN where “sequential” describes its consideration for a sequence of words.

4.1 The architecture of CNN:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_2 (Conv1D)            (None, 699, 1)            40001
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 349, 1)            0
_________________________________________________________________
flatten_2 (Flatten)          (None, 349)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 350
=================================================================
Total params: 40,351
Trainable params: 40,351
Non-trainable params: 0
_________________________________________________________________
```

In this architecture, word representations will be sent for the convolutions layer which has an output shape (batch size,699,1). In the Max pooling layer, it tries to shrink the output from convolutions with a pooling size of 2. So it gets reduced into half and has an output shape of (Batchsize,349,1), then it goes through flattening and dense layer (with 1 neuron) to get abstract features and produce classification as result.

5.Extension Parallel CNN

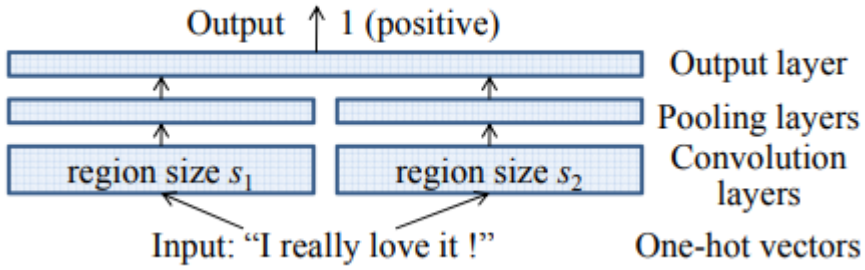
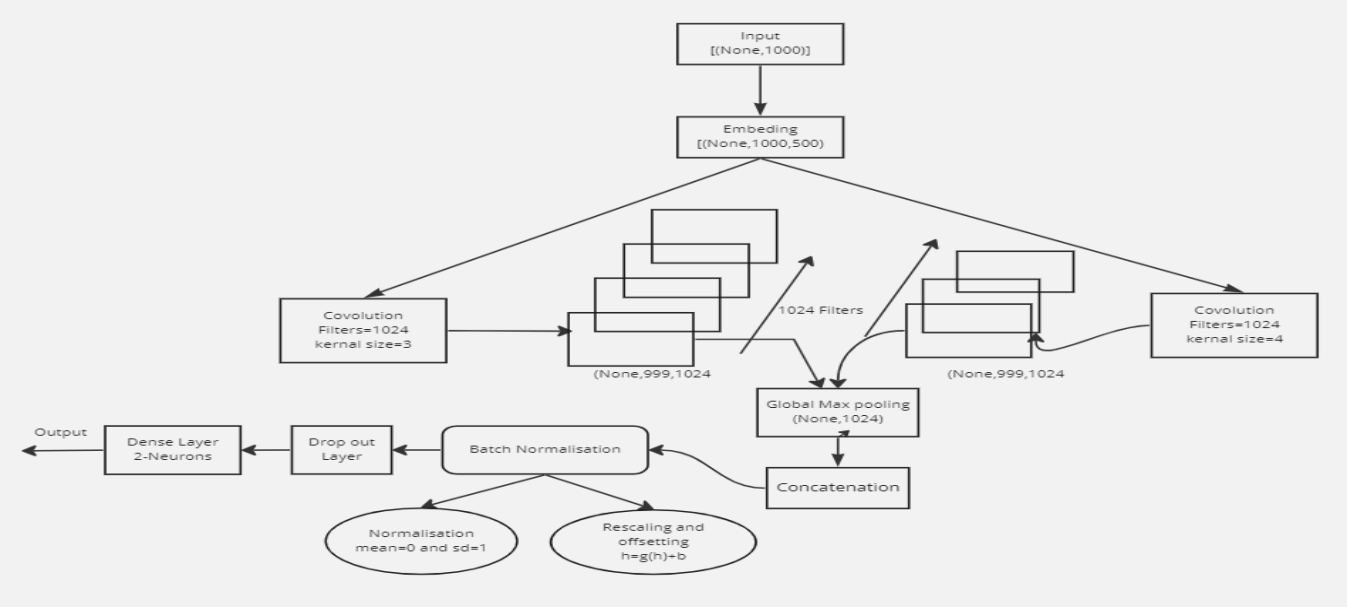


Fig 5(a)

So far, we have gone through CNN with single convolution and pooling layers. In this project, traditional CNN architecture was extended. In this extension multiple convolutional and pooling layers are added to run in parallel. The main idea involved in this approach is, for a given input region vector two pooling and convolution layer will perform their operation in parallel in which each convolutional layer has a different kernel size (Fig 5a). So, by doing this way we are feeding the same input data (One hot encoded vector) with 2 different region sizes to go through 2 convolution and pooling pairs in parallel. The topmost layer would concatenate the processed data from parallel pairs to generate classification.

5.1 Architecture of Parallel CNN

Initially, all the sentences are converted into constant size (1000) by padding zeros as sequences at the end. This is done to send it to the embedding layer because it required sentences with constant length to produce uniform results.



Word Embeddings:

To process the words in sentences it needs to be converted into some sort of number or vector form. This can be done using a bag of words, one hot encoded vector, etc. The problem with these approaches is when two similar-meaning sentences are converted into vectors their vector representation will not be similar, they look completely different this will not provide optimal solutions for NLP problems especially while performing sentiment analysis. This problem was resolved by the word embeddings layer which tries to create similar vector representations for nearly related sentences.

Hyperparameters in the embedding layer for the number of dimensions can be modified at the time of model creation. For larger volumes of data, a higher dimensional size needs to be provided. In this project embedding layer has a dimensional size of 50.

After going through the embedding layer input data flows through parallel pairs of convolutions and pooling layers with different kernel sizes in convolutional layers. Kernel sizes 3 and 4 were used for 2 convolutional layers with 1024 filters. For pooling, global max pooling was utilized for shrinking

1024 mapped images produced after convolutions. Then further in the architecture it concatenates both outputs from pooling pairs and goes through batch normalization.

Batch Normalization: it mainly involves two operations general normalization with mean 0 and standard deviation 1, second one is offsetting and rescaling in which it tries to slightly adjust the scale.

Batch normalization and Dense layer try to avoid overfitting which will eventually increase the performance while handling unseen data.

6. Overview of Data Sets and Data pre-processing steps.

In this project, 2 different data sets are utilized to understand the performance of CNN for text categorization using CNN.

Twitter Sentiment Data: This data set was trimmed due to limited computational capabilities. It contains tweets along with their respective sentiment label. This data set was obtained from Kaggle.

Spam Classification Data: It Contains mail along with its corresponding labels to identify whether the given mail content is spam or not. This Data set was obtained from Kaggle.

6.1 Data Pre-processing steps:

While dealing with NLP problems data cleaning is very vital before even building an actual model because it reduces unnecessary dimensional and computational strain on the GPU. Data Pre-processing steps utilized in this project are as follows

1. Removal of HTML tags

2. Removal of special characters and numbers

3. Removal of stop words

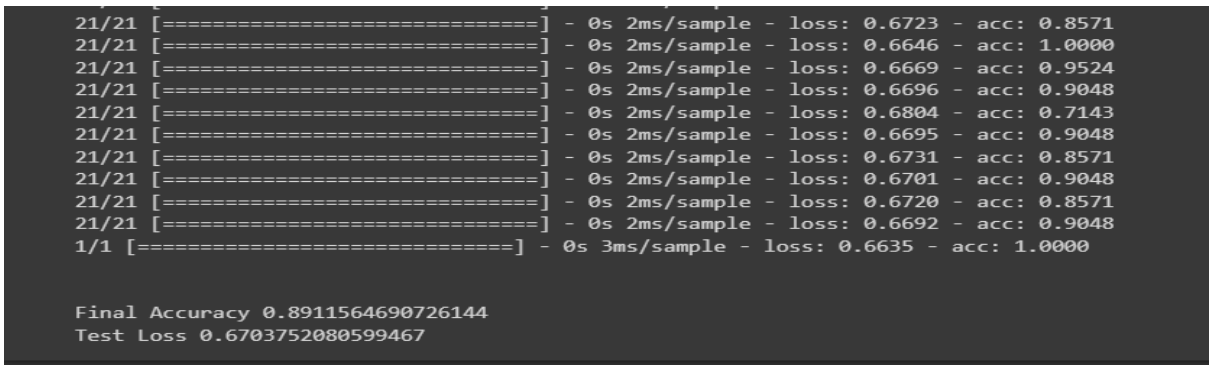
4. Padding the sequences to have the same length for all sentences

5. Tokenization of documents.

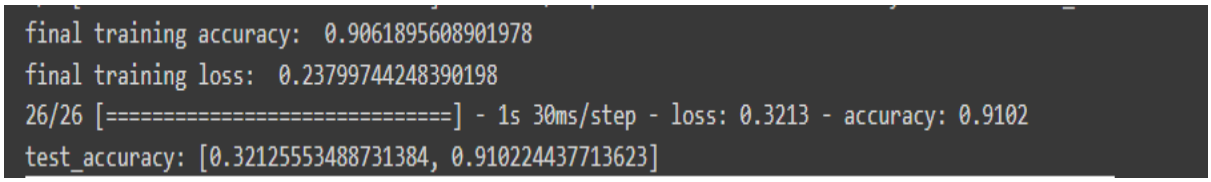
7. Results and Evaluation:

7.1 Spam Classification Data Results

Seq-CNN Results



Extension: Parallel- CNN Results



7.2 Twitter Sentiment Data Results

Seq-CNN Results

```
21/21 [==============================] - 0s 22ms/sample - loss: 0.5729 -
21/21 [==============================] - 0s 21ms/sample - loss: 0.5719 -
21/21 [==============================] - 0s 22ms/sample - loss: 0.5731 -
19/19 [==============================] - 0s 22ms/sample - loss: 0.5719 -


final test results
test_loss: 0.7023089671134949  test accuracy: 0.5009523809701204
```

Extension Parallel-CNN Results

```
final training accuracy:  0.8657456636428833
final training loss:  0.3171231150627136
63/63 [==============================] - 2s 28ms/step - loss: 0.6148 - accuracy: 0.5786
test_accuracy: [0.6147639155387878, 0.5785785913467407]
```

7.3 Comparison of Results

Data Set	Training Data Size	Test Data Size	Seq-CNN Accuracy	Parallel CNN Accuracy
Spam Topic classification data	5k mails	500	0.891	0.91
Twitter Sentiment Data	10k tweets	1000		0.578

Due to Computational constraints even after using google collab pro GPU, it could not be able to process more than 10k tweets which creates more than 10k high dimensional space for processing input in neural networks due to this constraint of limited data was utilized for tweets, neural networks for Twitter overfitted the data which resulted in poor accuracy while testing with 1000 unseen tweets but for spam topic classification data since the vocabulary is smaller in spam emails when compared with tweets, it provided better results for test data. Due to the parallel complimenting process of convolution and

pooling pairs Parallel CNN performed better than Seq-CNN in this project for spam topic classification and Twitter sentiments data.

8. Conclusion:

This project implementation provides an effective approach in which it can benefit from word order for text categorization using CNN. The methodologies implemented in this project provide an alternative to the traditional bag of words and traditional CNN for text approaches. The extension parallel CNN approach will be very effective while dealing with a large corpus of documents (millions or billions) because of its ability to learn from embeddings with different kernel sizes in parallel convolutional and pooling pairs. The main idea here is parallel convolutions and pooling layers complement each other to understand the input features better. In this project, good results were obtained for spam topic classification data but for Twitter data poor results were obtained due to medium scale data set with 10-15k tweets and a large vocabulary. Limited data was utilized because of the computational limitations even after using a google collab pro subscription. As mentioned in the paper extension of parallel CNN when tested on highly powerful GPUs with large-scale datasets with millions of records State-of-the-art performances are obtained for topic classification and sentiment classification.

9. References

[1] M. Kazanova, Sentiment140 dataset with 1.6 million tweets, 2017.

[Dataset]. Available: https://www.kaggle.com/datasets/kazanova/sentiment140.

[Accessed: October 17, 2022].

[2] Balaka biswas, Email spam classification , 2019. [Dataset]. Available: https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv [Accessed: October 17, 2022].

Conclusion