Paper Title\* **(**use style: ***paper title***)

Subtitle as needed ***(paper subtitle)***

Alec Yenter

*College of Engineering and Computer Science*

*California State University*

*Fullerton, California*

*alecyenter@csu.fullerton.edu*

Abhishek Verma, Ph.D.

*College of Engineering and Computer Science*

*California State University*

*Fullerton, California*

*averma@fullerton.edu*

*Abstract*—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. *\*CRITICAL: Do Not Use Symbols, Special Characters, or Math in Paper Title or Abstract*. (*Abstract*)

Keywords—component; formatting; style; styling; insert (key words)

# Introduction

Introduction – .5 pg

# Related Work

To build a background to the paper, the following sections review related literature of neural networks and text classification.

## Neural Networks (NN)

Groundbreaking progress in machine learning has been made through the use of neural networks. The ability for a neural network to imitate the brain’s processes is a valuable method for problems in textual, visual, and other signals. The basics for neural networks are layers each with a specific number of nodes. Input data is the first layer and connections are given to the next layer. This is repeated until the final layer, where the output is produced. There are multiple types of networks and layers; in this paper, we will specifically use three layers: Fully-Connected Layers, Convolutional Layers, and LSTM layers.

### Fully-Connected (Dense) Layers are simple layers where every neuron from the previous layer is connected to every neuron in the dense layer. These layers can do basic learning while reshaping the input data, for instance reducing the output to one neuron.

### Convolutional Neural Networks (CNN) are a specific type of neural network that can work well with spatial data. CNNs are particularly useful with images for tasks such as classification. Convolutional layers use only certain connections from previous layer; specifically, local neurons are connected to the neurons of the next layer. This method causes the layer to gain more of an understanding of the genreal view of the inputs.

### Long Short Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). RNN neurons have a connection to the previous neuron state in addition to the layer inputs. RNNs are extremely benefitial to data that is sequential or that can value a contextual view. LSTMs are a formof RNNs where newer information in the neurons is more critical than older information. LSTMs are useful on sequential text data, because, while the context of a parargraph and/or a sentence is considered, the most recent words hold the most weight on the current neuron state.

## Neural Network Text Classification

When given textual data, a typical objective is to classify the information for analytical or statistical purposes. [2] compared multiple n-gram machine learning approaches on the same IMDb review sentiment dataset used in this paper. The data was preprocessed before being vectorized and fed into various configuration of machine learning algorithms. Many of these techniques reached high accuracies in the 80% with the best configuration “Unigram + Bigram + Trigram” reaching the maximum accuracy of 88.94%.

While there are many useful mathematical algorithms, this paper focuses on neural network approaches to textual classification. These approaches have been split into character-level classification and word-level classification for the purposes of this paper.

### Character-Level Text Classification is a newer approach that focuses on the letters of the text. This approach has the benefit of avoiding the need for a dictionary or an understanding of the language, but instead defines an albhabet for the data. [4] explored the use of charater-level convolutional networks for classification of textual documents. The use of the sequence of letters as a signal sequence produced comparable results to other methods such as word-level classification. [4] developed a neural network of nine layers: six convolutional layers followed by three fully-connected layers. These layers were accompanied my max-pooling layers to help the nework handle the nine-layered depth.

[6] built onto character-level text classification by using a CNN to produce a form of word embedding from the character inputs of a word. For each word, letter embeddings are concatenated along axis 1, convoluted through multiple kernels, and finally pooled and concatenated into a flat layer. The last layer produces a representation of the word that is fed through a highway network to a LSTM network for classification. The benefit of this hybrid method is that no dictionary is needed, yet the network learns on a word-level basis that shows a human-like understanding of the language.

### Word-Level Text Classification is a more traditional method to text classfication with neural networks. [5] created a shallow CNN that classifies using multiple different kernels. In this network, a convolutional layer will look at n words at a time when applying the filters and pooling-over-time. The network would use multiple kernel sizes and concatenate the results; therefore, the network finds context from the n number of words nearby the word. The final connected layer uses this information for classification. This network is simple but effective in textual classification.

[1] uses IMDb review data with a new LSTM neural network to classify sentiment of the review. The review data contained no neutral data. The dictionary was limited to the top 2000 most used words and each review sequence was capped at 100 words and padded with zeros if less than the max. The proposed LSTM layer is a biologically-inspired additive version of a traditional LSTM that produced higher loss stability, but lower accuracy. The best accuracy achieved between both LSTM models was still under 85%. The use of an LSTM on textual data gives better contextual view of words than a CNN.

# Description of Dataset

The ACL Internet Movie Database (IMDb) dataset used was created in [3] for learning word vectors. The dataset consists of 100,000 textual reviews of movies; half (50,000) of the reviews are for testing and have no label. The other (50,000) reviews are paired with a label of 0 or 1 to represent negative and positive sentiment, respectively. These labels were linearly mapped from the IMDb’s star rating system where reviewers can rate a movie a certain number of stars from 1 to 10. The labeled reviews are split 50/50 into training and validation data. Each set has 12,500 positive reviews and 12,500 negative reviews to keep the data balanced.

There is at most 30 reviews for any one movie. Each review can have anywhere from min to max number of words. The mean number of words is 234.76 with a standard deviation of 172.911495 words. Certain punctuation and symbols are included, such as “?”, “!”, and “:(“. Collectively, the dataset contained 88,585 different words across all reviews.

The deep learning library Keras provides a simple import method to retrieve these reviews in a preprocessed format. The function grabs the reviews and encodes them into a sequence of word indices according to a dictionary of the *D* most frequently used words in the dataset, where *D* is given. For further flexibility, the indices are ordered by the frequency of the words, such that the word mapped to index 2 is the 2nd most common word in the dataset. Index 0 is reserved for unknown words that are not in the dictionary. The reviews are then padded to fit a desired maximum sequence length; longer reviews are truncated and shorter reviews are padded with zeros. As a final step, the first layer of the neural network converts the indices to embeddings of *E* dimension.

For this paper’s experiments, the dictionary was capped at 5,000 words with a maximum padded sequence length of 500 words.

# Methodology

The proposed method in this paper utilizes a CNN and a LSTM on word-level classification of the IMDb review sentiment dataset. The method combines versions of the networks from [5] and [1]; the proposed network has multiple branches that accept the data and perform convolution. The output of the CNN layers is fed into an LSTM similarly to the method proposed in [6] before being concatenated and sent to a fully-connected layer in order to produce a single, final output. The network is trained and tested in mini-batches between 16 and 64 to quicken the process.

## Embedding

The first layer of the network accepts the input reviews as a sequence of indices and embeds each word into a vector of a specific size *e*. (e.g. a vector of 500 word indices embedded at 32 becomes 500 vectors of length 32). The embedding layer is a matrix of trainable weights that, through matrix multiplication, produce the vectors for each word index. Therefore, during training, the embedding layer improves on the embeddings of each word.

## Convolution

The output of the embedding layer is given to each branch of *b* branches. Each branch starts with a 1-dimensional convolution layer of kernel size *c* specific to that branch. The kernel during 1-dimensional convolution is of shape kernel sizeby embedding size (*c* x *e*). The kernel will then preform convolution on whole words instead of the typical 2-dimensional convolution that would filter partial widths and cut words into pieces. The layer produces multiple outputs with the use of multiple filters *f*.

The purpose of the CNN layer is to view word combinations of the kernel size *c*. The result is an understanding of words when used with other words. For example, when *c=3*, the layer views 3 words at a time and, therefore, establishing a sense of 3-word combinations. This layer’s output has a shape of input height by filters (*words* by *f).*

## Activation

Each branch applies a rectified linear unit (ReLU) activation of the CNN layer’s output; this layer replaces any negative outputs with zero. The ReLU layer is used in order to introduces non-linearity into the network. The output of this layer is the same shape as the input shape.

## Max Pooling

Each branch undergoes 1-dimensional max pooling following ReLU activation; this layer converts each kernel size of the input into a single output of the maximum observed number. The result is a reduced, down-sampled version of the input. The purpose of the layer is to reduce overfitting, while allowing for further processing. Similarly to the CNN layer, the 1-dimensional max pooling kernel shape is fit to the width of the data, so that the parameter kernel size *p* implies a kernel shape of *p* by data width. This technique allows pooling to be applied with the understanding that data is composed of whole words. The output of this layer is a reduction in height according to kernel size *p* (input height ÷ *p*) .

## Dropout

After max pooling, each branch goes through a dropout layer; this layer randomly sets a portion of the inputs to 0. The dropout is applied to specified *d* fraction of the inputs. This layer serves to prevent overfitting and generalize the network to not focus on specific pieces of input. The output shape is equivalent to the input shape.

## Batch Normalization

The next layer for each branch is batch normalization; this layer simply normalizes the distribution for each batch after dropout. The purpose of batch normalization is to reduce internal covariate shift and therefore lead to convergence at a faster rate. The output of this layer holds the same shape as the input.

## LSTM

The final layer for each branch is a LSTM layer with a specified number of units *l*. The LSTM is used because of the nature of sequential data. The layer’s persistence allows knowledge of previous input (convoluted word combinations) to influence subsequent input. The output has a length of the number of units *l*.

## Concatenation

The branches are finally merged together through concatenation. The LSTM layers’ outputs are combined together in an array. The output shape of this layer is equal to the summation of the output of all the branches (*l* x *b*) .

## Dense

The last layer is a fully-connected layer from the concatenated input to a single output. The layer is followed by a simple sigmoid activation function to conform the output between 0 and 1. The final yield is a single output.

## Binary Crossentropy

The network is compiled with a binary crossentropy loss function; this loss calculates loss with two classes (0 and 1). For this paper’s purposes, 0 represents negative sentiment and 1 represents positive sentiment. The loss is calculated on the single and final output of the dense layer.

# Experiments

Many experiments were attempted to determine appropriate parameters. The IMDb review sentiment dataset was used for all experiments. The dataset was preprocessed to a dictionary size of 5,000 words with a zero-padded maximum sequence of 500 words per a review; anymore data became insignificant to the networks objective. The following sections establish the best parameters approximated by the experiments.

## Embedding

An embedding size of 32 best fit the dataset. Attempts to use GoogleNews and Wikipedia pre-trained embeddings (as untrainable) proved to have no positive effect.

## Convolution

The largest alterations where determined by the number of branches to be used; this was directly tied to the kernel sizes used. The optimal kernel sizes where two through seven; therefore, six branches. Viewing two through seven words at a time proved to extract the most critical information and produce higher accuracy. The optimal kernel size was found to be kernel\_size.

Additionally, the convolutional layer was accompanied by a ridge regression (*l2*) kernel regularizer to help prevent overfitting. The *l2* parameter was set to 0.01.

## Activation

The ReLU activation layer proved crucial to preventing overfitting. This layer has no parameters.

## Max Pooling

Although max pooling assisted with the chief issue, overfitting, large kernel sizes proved to decrease accuracy. The optimum kernel size during the experiments was 2; reducing the height of the input by half.

## Dropout

The dropout layer was recognized to be the best option to reduce overfitting. The dropout was set at a rate of 0.8 to force other weights to help generalize the network. While slowing convergence, this method led to higher accuracy and a better understanding of the data.

## Batch Normalization

Batch normalization was added to the network to help with overfitting with the LSTM network. This layer has no parameters.

## LSTM

Each branch’s LSTM layer has 100 units. Any less or more units reduce accuracy or increase overfitting.

## Concatenation

The merging layer has no tuning parameters.

## Dense

The final dense layer has no tuning parameters.

# Results and Analysis

Results and analysis – 1 pg

# Conclusion

Conclusion - .5 pg

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g.” Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi resolution TIFF or EPS file with all fonts embedded) because this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

1. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
2. I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
3. K. Elissa, “Title of paper if known,” unpublished.
4. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
5. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
6. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.