Character Level Convolutional Network for Text Classification

Team Number: 23
Team Members:
Harish Umasankar - 2020102067
Bodgam Rohan Reddy - 2020102039
Ruhul Ameen - 2020102031
Akash Vallamsetty - 2020102050
Github Repo: Link

Introduction

Text classification is an important problem in Natural Language Processing. It involves categorizing text into organized groups. The accuracy and performance of text classification can be improved by selecting the best features and best classification methods.

Since our goal is to classify text, words are the most commonly used unit for extracting features. But with recent advancements in Neural Networks, we can directly use characters to extract features.The features obtained from the characters are fed to the neural network.

Goal

Our goal is to develop the Character Level Convolutional Neural Network proposed in the paper. This involves feature extraction using One Hot Encoding to create a character embedding and feeding the features to the Convolutional Layers followed by the Fully Connected Layers.

We implemented various other models to compare the performance of the proposed model against the various

- Traditional Models
    - Multinomial Regression using n gram TFIDF

    - Multinomial Regression using Bag of Words TFIDF
- Deep Learning Models
    - LSTM

We have also analyzed the model on the basis of the following
- Dataset Size
- Character Quantization
- Overall Effectiveness of Character Level CNN
- Application of Free Lunch Theorem

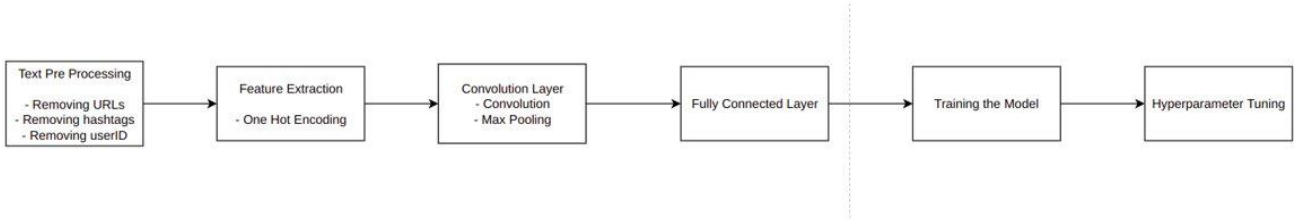
Dataset

Dataset used for testing all the models is the collection of 1.6 million tweets which are either annotated to be positive or negative.
The dataset is divided accordingly to train and test the model.
Link to the dataset : dataset

Models Developed

Model 1: Character Level Convolutional Neural Network

Pipeline



Pre-Processing

The raw tweets data consist of user IDs, URLs, and hashtags. Since URLs and user IDs do not have any information valu, we first remove those words/strings from each of the tweets.

Hashtags carry some meaning because people use hashtags that are related to the tweet. Therefore whenever we get a hashtag with some phrase after it, we remove only the hashtag and retain the phrase after it.

Since the character quantization consists only of lower-case alphabets and does not consist of upper-case alphabets, we convert all upper-case alphabets into lower-case.

Eg: Tweet: “This is just DIRTY politics @rahulG <img URL>”

Here we replace @rahulG as rahulG and the image URL is completely removed.

After pre processing,
Tweet: “this is just dirty politics rahulG”

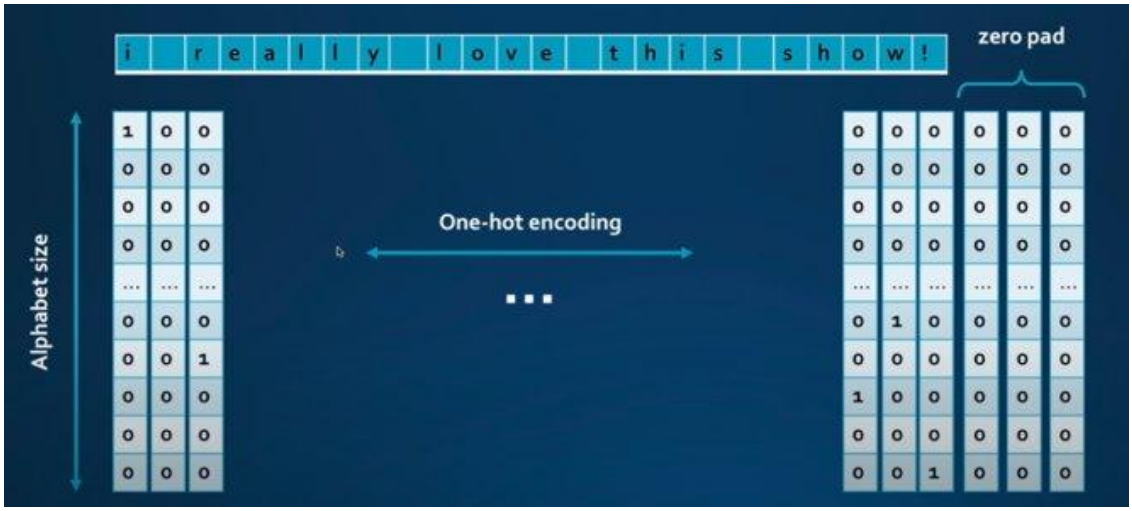
Feature Extraction Technique: One Hot Encoding

The characters in the text are quantized into 70 symbols - 26 English letters, 10 digits, 33 other characters and the new line character . The list of symbols are as below

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:’’’/\|_@#$%^&*~‘+-=<>()[]{}
```

Since there are 70 characters, we have 70 binary variables. With one-hot, we convert each character into a new categorical column and assign a binary value of 1 or 0 to those columns. Each character in a word is represented as a sparse binary vector. All the values are zero, and the index corresponding to the character is marked with a 1.

One Hot Encoding is useful when the data we are using is categorical and does not have any ordinal relationship.



one hot encoding representation

CNN Architecture

The CNN Network takes the character embedded - one hot vector as an input.

The CNN Network is 9 Layers deep and consists of the following components

- 6 Convolutional Layers
- 3 Fully Connected Layer.

Convolutional Layer

The input has a number of features equal to 70 due to our character quantization method, and the input feature length is 1014.

The key modules of the convolutional layer are as below

Strided Convolution

The convolutional layer applies a kernel to the input to create a feature map that consists of the features detected from the input.

$$h(y) = \sum_{x=1}^{k} f(x) \cdot g(y \cdot d - x + c),$$

Here, f(x) represents the kernel and g(x) represents the input to the convolutional layer. Value c represents the offset and d represents the stride value.

Temporal Convolution

Each convolution layer is followed by a max-pooling layer. It is used to reduce the feature vector dimension and extract the most present features.

$$h(y) = \max_{x=1}^{k} g(y \cdot d - x + c),$$

Here g(y) represents the input and d represents the stride value and c represents the offset. The maximum is obtained from a window of size k.

Activation Function

The activation function is a function that fires the node based on a threshold value.

We did not use sigmoid or tan H function due to vanishing gradient problems. The activation function used is the reLu function h(x) = max(0, x) as it does not suffer from the vanishing gradient problem.

Fully Connected Layer

The Fully-connected layers connect every neuron in one level to every neuron in the next level. The model consists of 3 Fully Connected Layers

Model 2: n-gram TFIDF with Multinomial Logistic Regression

TFIDF:

Term-frequency Inverse-document-frequency is a technique used to convert text into vectors with the help of the occurrence of keywords in the samples. The technique uses the number of occurrences of a word in the sample(a single tweet) and the whole dataset(collection of tweets) and finds the significance of words in the document.

Term-frequency calculates the frequency of the word in the whole dataset.

$$TF(word) = \frac{NO.\ of\ occurrences\ of\ word\ in\ the\ sample}{Total\ No.\ of\ words\ in\ the\ sample}$$

Term-frequency is basically the probability of the word in the document which always lies in range [0,1]. Term-frequency of a word is different for different samples.

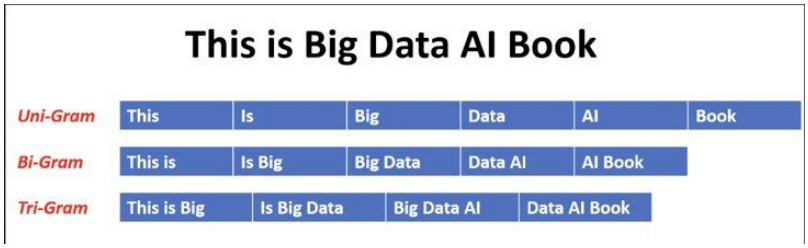
Inverse-Document-Frequency is the measure of the relevance of the word in the document.

$$IDF(word) = log(\frac{Total\ No.\ of\ samples\ in\ dataset}{No.\ of\ samples\ containing\ the\ word})$$

IDF of a word can lie in range [0,∞). IDF is same for a word in the whole dataset.

TF-IDF for a word in the dataset is the product of TF(word)*IDF(word). The results will give the significance of the word in the dataset.
TF-IDF in most cases are normalized to give proper results.
Lower the TFIDF, higher the occurrence of the word in the dataset.

n-gram is the sequence of n consecutive words in the tweet.



Multinomial Logistic Regression is used to classify the dataset. Multinomial Logistic Regression is used in cases where there are multiple classes to classify the model into. It is done by developing K-1 logistic regression models between two classes for K classes and predicting the most frequent occurring class for a sample in K-1 models.

The model developed uses the 500000 most frequent n-grams ranging upto 5-grams (5 consecutive words in tweet) and calculates the TFIDF of each possible n-gram. Using the feature vectors obtained, we classify the tweet as positive or negative using multinomial logistic regression.

Model 3: Bag of Words TFIDF with Multinomial Logistic Regression

Bag of words normally considers the frequency of words as features but they do not contain information on the relevance(semantics) of the words in the dataset. So, TFIDF is used to find the significance of each of the words in the dataset with TF being the frequency of word and IDF being the value signifying the occurrence of word in different samples. Then with the help of multinomial regression, a classifier is developed to classify the tweets.

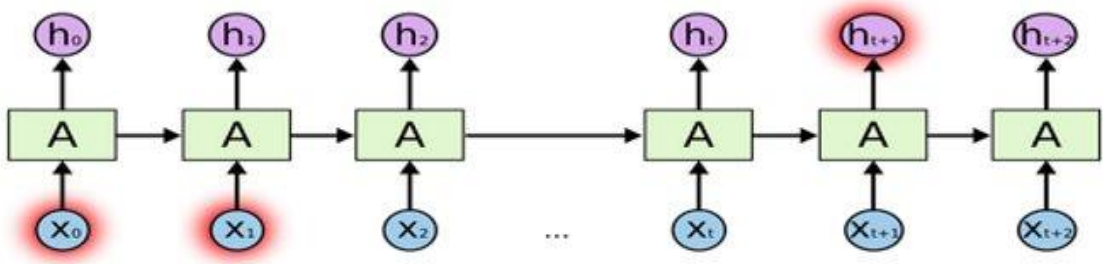
Bag of words for a given statement

The model developed uses the 50000 most frequent words in the dataset to calculate the feature vectors using TFIDF and with the help of

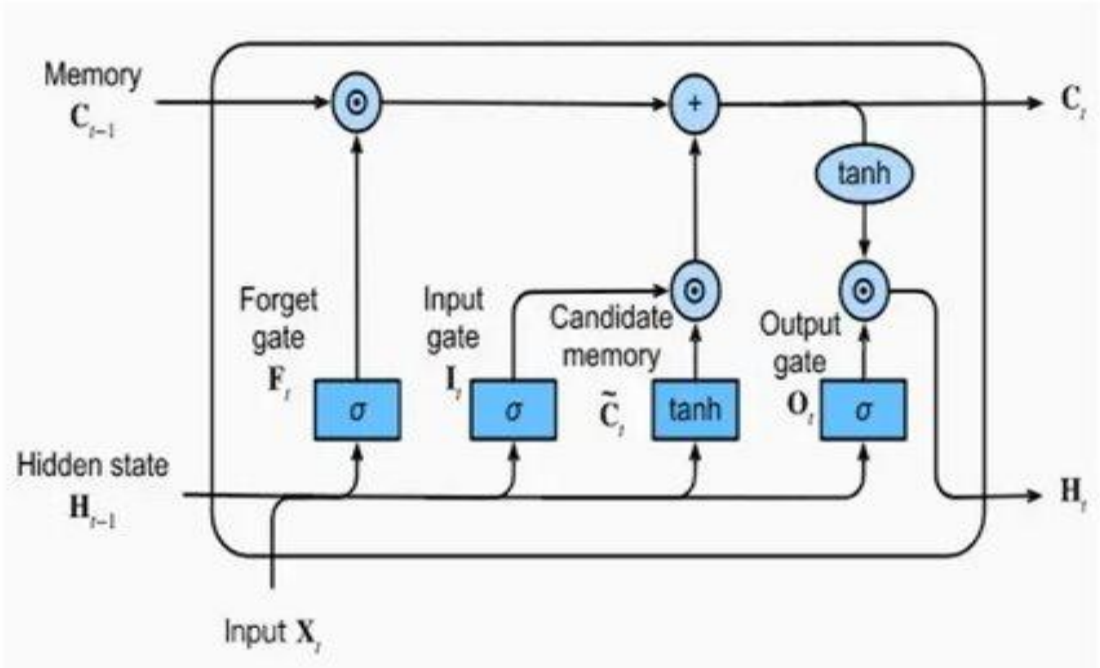
multinomial logistic regression, a classifier is developed for sentiment analysis.

Model 4: Long Short-Term Memory

In recurrent neural networks, A feedback loop is present to ensure that the prediction of the current item depends on the previous series of events. In text classification, It is very required as texts normally have meaning only when it is written in a certain sequence. So, RNN take cares of this thing to provide better results. But normal RNN fails to connect the words which has a large gap but very relevant to current prediction. Eg., “I grew up in France,... so I speak fluent French”. In this example, the prediction of the word “French” heavily depends on the word “France” which has large gap. In this case normal RNN fails.



LSTM solves this issue by maintaining a long term memory stream along with a short term memory stream. LSTM uses two types of activation functions - sigmoid function, tanh function. LSTM operates through three steps involving gates, First one is Forget gate which decides upon the long term memory to retain based on current input and the last short term memory. Second one is the input gate which updates the long term memory. The third is output gate which predicts the output based on both the long term and short term memory streams which is also the next short term memory.



Using the mechanism, a model is developed which converts all the words to vectors. Then the average of outputs of all such LSTM cells are considered as feature vectors which thereby are used to train a classifier using multinomial logistic regression.

Training and Testing Methodology

The dataset consists of 1.6 Million Tweets. Using all the tweets for text classification would require a lot of RAM to process. Hence we have used only 30% of the dataset and split this 30% into 80% for training and 20% for testing.

We divide the training dataset batch wise and run multiple epochs for each batch.

Results

(table 1: Comp acc of all 4)

Model	Precision	Recall	F1-measure	Accuracy
CNN (positive)	0.85	0.81	0.83	0.84
CNN (negative)	0.82	0.86	0.84	
N-gram TFIDF (positive)	0.78	0.81	0.79	0.79
N-gram TFIDF (negative)	0.80	0.77	0.78	
Bag of words TFIDF (positive)	0.77	0.80	0.78	0.78
Bag of words TFIDF (negative)	0.79	0.75	0.77	
LSTM (positive)	0.75	0.79	0.77	0.77
LSTM (negative)	0.78	0.74	0.76	

(table 2: accuracy for CharCNN with diff dataset size)

Model	10 percent dataset	30 percent dataset
Character-level CNN	0.76	0.84
N-gram TFIDF	0.76	0.78

(table 3: accuracy for using UpperCase and Lower Case vs Only LowerCase)

Model	Precision	Recall	F1-measure	Accuracy
Lower case CNN (positive)	0.85	0.81	0.83	0.84
Lower case CNN (negative)	0.82	0.86	0.84	
Lower and Upper case CNN (positive)	0.76	0.74	0.75	0.75
Lower and Upper case CNN (negative)	0.75	0.77	0.76	

Analysis

Effectiveness of Character Level CNN

From table 1, we can observe that the accuracy is highest for Character Level CNN compared to other Traditional and Deep Learning Methods.

This gives a strong indication that text classification can be done by knowing the alphabets of any language. We do not need the knowledge of the dictionary of all words in that language.

Dataset Size

From table 2, we can observe that as we decrease the number of samples in the dataset, the accuracy of the Character Level CNN decreases.

We can also observe that the accuracy of the TFIDF Traditional Model is greater than the accuracy of the proposed model.

The reason for this is that Deep Learning Models require a large data set to learn and train the model, unlike Traditional Models where there is no learning involved.

Choice of Alphabet

In table 2, we have compared the accuracy obtained by quantizing using only Lower Case Alphabets against quantizing using Upper Case and Lowercase Alphabets.

We observe that quantizing using Lower case alphabets performs better. The reason for this can be because we predict the sentiment according to the meaning of the word and not the semantic. Quantizing with upper and lower case alphabets would mean that we are telling the model that “Happy” and “happy” must be treated differently. This is equivalent to providing wrong information to the model. As a result quantizing with upper and lower case has poor accuracy compared to quantizing using lower case alphabets.

Free Lunch Theorem

Free Lunch Theorem states that “There is no single model that works best for every situation.”

Our experiments verify the above statement as the accuracy of classification depends on the data set type, data set size, quantization and other features.

Work Distribution

1. Harish Umasankar
    a. Character Level Convolutional Neural Network (as proposed in paper)
    b. Analysis of the Model by comparing the effect of each parameter - size, alphabet casing etc
2. Rohan Bodgam Reddy
    a. n gram TFIDF + Multinomial Logistic Regression Model
    b. Pre-processing
3. Akash
    a. Bag of words TFIDF + Multinomial Logistic Regression Model
4. Ruhul Ameen
    a. LSTM Model

Link to Notebooks/Code

- Character Level CNN using 30% dataset - Quantized using Lower Case
- Bag of Words and its TFIDF using 30% dataset
- N-grams and its TFIDF using 30% dataset
- LSTM using 30% dataset
- Plots comparing the accuracies

Experimentation

- Character Level CNN - Quantized using Upper and Lower Case
- Character Level CNN using 10% dataset
- N-grams TFIDF with 10% dataset

Note: Dataset is taken as input in different codes in different ways, so it should be checked and given accordingly.

Conclusion

From the paper and the analysis, we can conclude that the Character Level Convolution Method is an effective method and can be used for Text Classification problems when we have a large dataset. But it is also important to note that in cases where dataset size is limited, the traditional methods work better.