Final Project report: CREPE

Team Members	Roll Number
Rohan Asokan	2019101031
Karthik Viswanathan	2019113015
Abhishekh Sivakumar	2019101014
Satyakameswar Rao	2021900005

Introduction

This project aims to construct a character-level convolutional net (CREPE) and test it on the Yelp Polarity dataset. We benchmark our model with other baselines such as Bag of Words and LSTMs. We then explain the CREPE model's robustness theoretically and describe why the CREPE model performs better than Bag of Words and LSTMs.

Dataset

To implement this, we have used the Yelp reviews dataset obtained from the Yelp Dataset Challenge in 2015. This dataset contains 1,569,264 samples that have review texts. Two classification tasks are constructed from this dataset – one predicting the total number of stars the user has given and the other indicating a polarity label by considering stars 1 and 2 negative and 3 and 4 positive. The full dataset has 130,000 training samples and 10,000 testing samples in each lead, and the polarity dataset has 599,999 training samples and 19,000 test samples in each polarity.

We have primarily used the polarity dataset, a two-class classification problem. Either a positive review or a negative review is the target parameter.

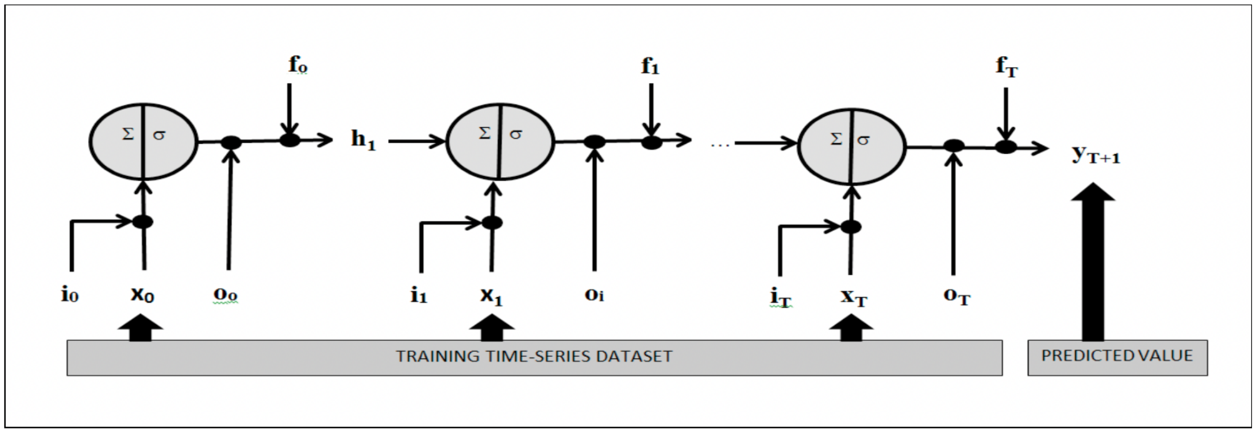
Baselines

From the many baselines experimented by the paper's authors, we pick up two baselines: Bag of Words (Linear Naïve Bayes two-class classifier) and LSTMs. Using PyTorch, we construct these from scratch and feed them into the vectorized dataset.

LSTM

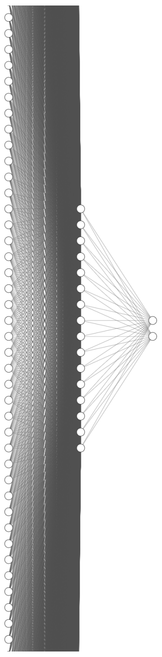
LSTM stands for Long-Short Term Memory. LSTMs perform better compared to traditional RNNs. Like NN, LSTM can have multiple hidden layers, and as it passes through every layer, the relevant information is kept. All the irrelevant information gets discarded in every single cell. We have implemented a standard single-layer LSTM network. Vectorizing for the dataset is made using TFIDF. We remove the first n words as they correspond to stop sequences.

Architecture for the LSTMs have been provided below:



Bag of Words

A bag of words is a method of simple feature formation for textual data. Text is something that machine learning models cannot quickly parse. To combat this bag of terms was developed as a statistical formation for numerical vectors. Initially, a corpus-based on tokenization is formed. A simple one-hot vector can be assigned for each token or sentence accordingly. A simple extension to this would be the addition of other statistical properties, including document frequency of a term and global frequency of a time, also known as TF-IDF. We now pass the tokenized dataset (using count vectorizer) into a simple feed-forward architecture as follows:



The outputs are characterized by taking the maximum argument outputted by the two neurons. We train this model for 10 epochs.

Character-level Convolutional Networks (CREPE)

Crucial concepts from the paper are listed below. All other techniques used for training are pretty common in machine learning research. The usage of temporal convolutions and character quantization makes this paper novel.

Temporal Convolutions

The temporal convolutional module computes a 1-D convolution. This is a prime part of the CNN paper.

The convolution $h(y) \in [1, b(l-k+1)/dc] \rightarrow R$ between $f(x)$ and $g(x)$ with stride d is defined as:

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

The key module to train deeper models is temporal max-pooling:

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

Given a discrete input function $g(x) \in [1, l] \rightarrow R\, g(x) \in [1, l] \rightarrow R$, the max-pooling function

$h(y) \in [1, \frac{b(l-k+1)}{dc}] \rightarrow R$ of g(x) is defined as

$$h(y) = (max(x = 1, k))g(y \cdot d - x + c)$$

Character Quantization

The developed models accept a sequence of encoded characters as input.

It is done by prescribing an alphabet of size m for the input language and then quantizing each character using 1-hot encoding. Then, the sequence of characters is transformed into a series of such m sized vectors with fixed length L0. Any surface exceeding length L0 is ignored, and any characters not in the alphabet, including blank symbols, are quantized as all-zero vectors.

The character quantization order is backward. The latest reading on characters is always placed near the beginning of the output, making it easy for fully connected layers to associate weights with the latest task.

Small and Large network

We distinguish the character level CNN's constructed in the paper as small and large models. The difference between these models is the number of channels and the train time. The accuracies for these models vary by a small amount, as presented in the later sections. The network architectures for both have been provided below:

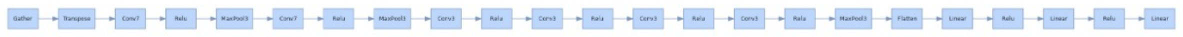
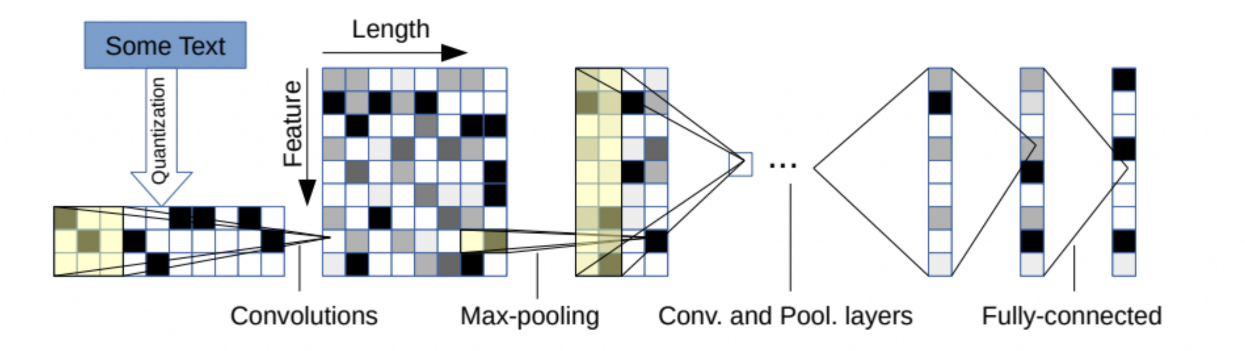


Illustration of the model is as follows:



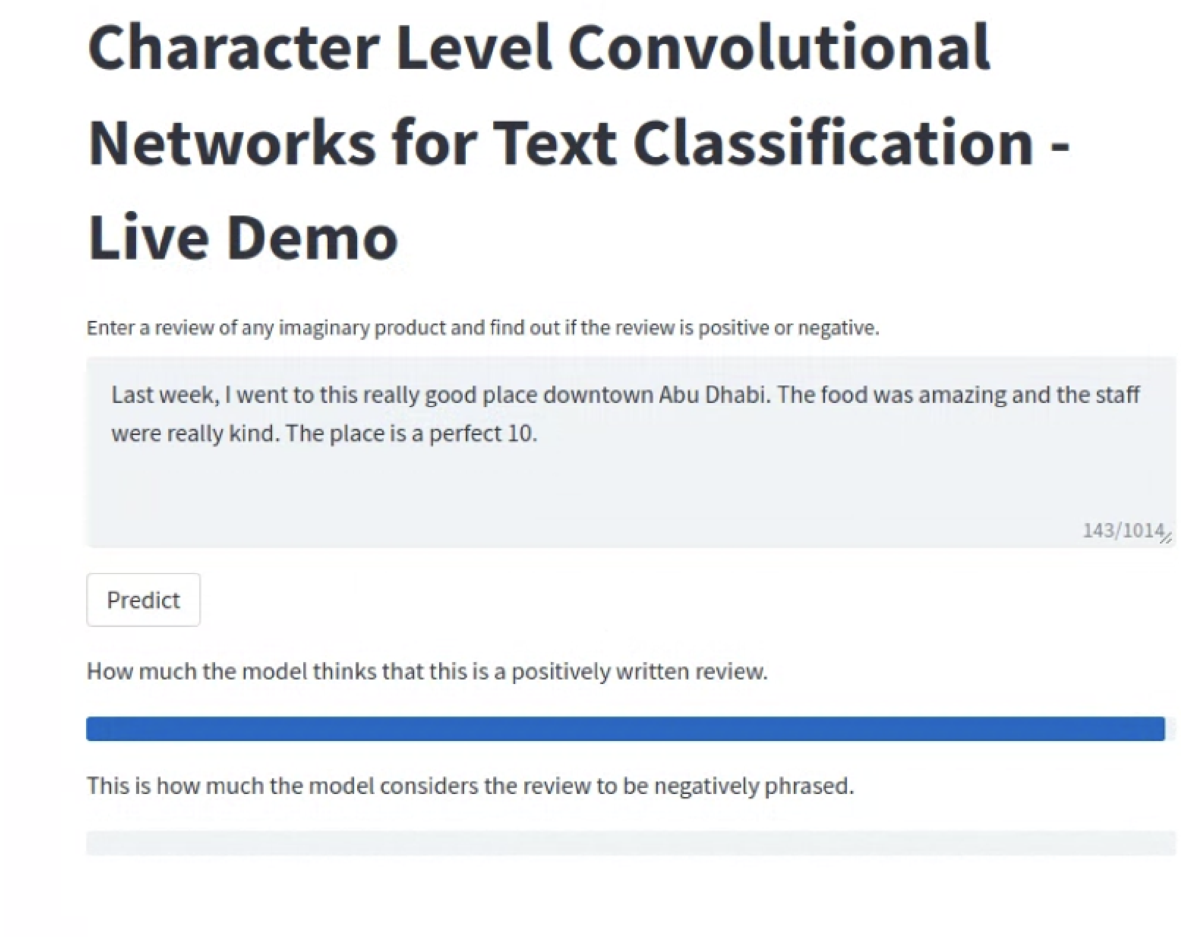
Results

Model	Validation Accuracy	Loss Function	Loss	Epochs
Bag of Words	91%	Cross-Entropy	0.16	10
LSTM	89.53%	Cross-Entropy	0.28	5
CREPE-Small	93.76%	Cross-Entropy	0.146	6
CREPE-Large	94%	Cross-Entropy	0.13	6

Live Demo

We constructed a live demo of our model using streamline. We have pasted some sample outputs over here:

A positive review



Character Level Convolutional Networks for Text Classification - Live Demo

Enter a review of any imaginary product and find out if the review is positive or negative.

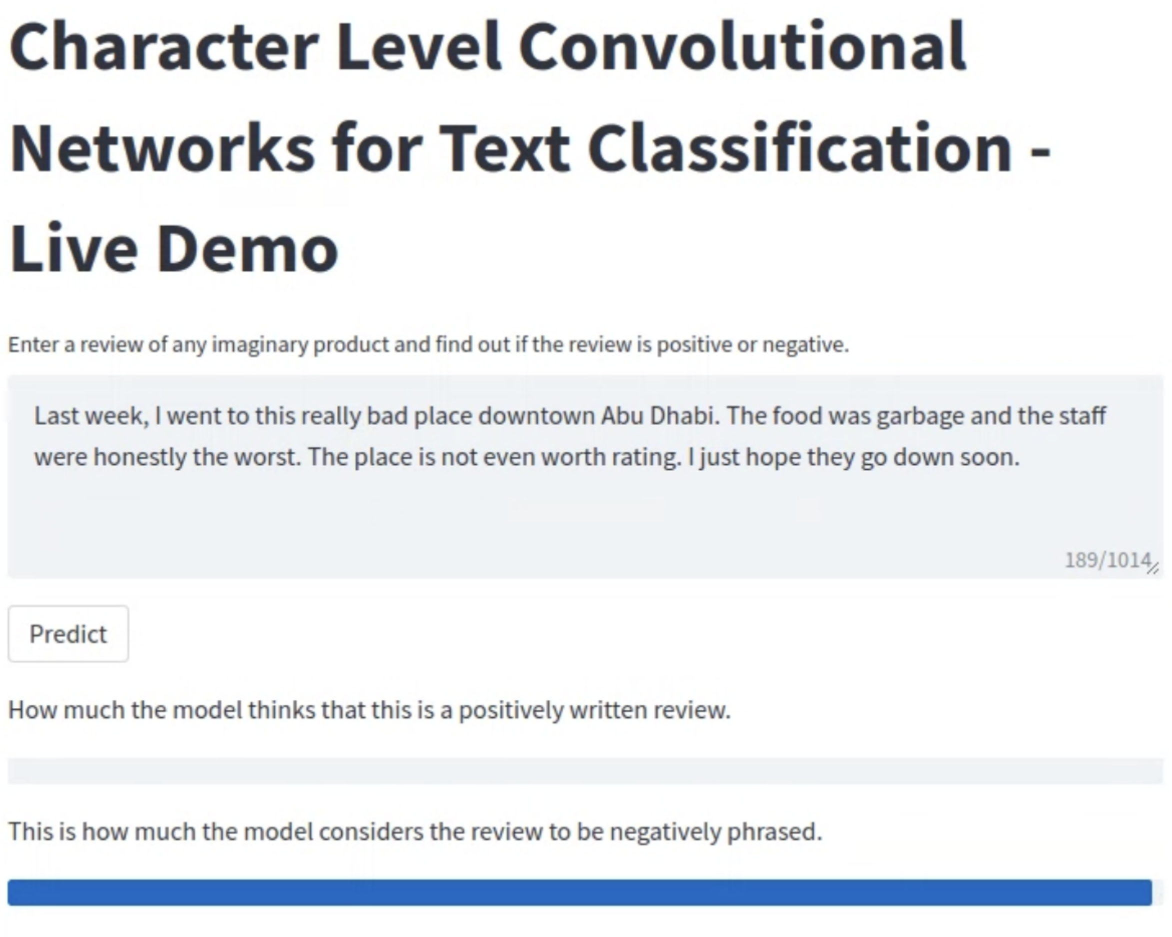
Last week, I went to this really good place downtown Abu Dhabi. The food was amazing and the staff were really kind. The place is a perfect 10.

143/1014

How much the model thinks that this is a positively written review.

This is how much the model considers the review to be negatively phrased.

A negative review


Character Level Convolutional Networks for Text Classification - Live Demo

Enter a review of any imaginary product and find out if the review is positive or negative.

Last week, I went to this really bad place downtown Abu Dhabi. The food was garbage and the staff were honestly the worst. The place is not even worth rating. I just hope they go down soon.

189/1014

How much the model thinks that this is a positively written review.

This is how much the model considers the review to be negatively phrased.

Key takeaways

1. The character level classification network is an effective model. In this work, the authors present a very fine method to conduct text classification using convolutional networks. **Language can also be thought of as a signal no different from any other kind.**

2. ConvNets work very well in real-time user-generated data. Depending on the text curation quality and the similarity in training and testing domains, the model has been very effective in predicting real-time data.

3. According to the authors, the models are case-sensitive. The authors are yet to validate that case sensitivity induces some kind of regularization.

4. Be it sentiment analysis or topic classifications, ConvNets can be used in a variety of prediction tasks.

Code and Dataset

The code for this project can be found at abhishekh2001/SMAI-Project: Implementation of Character-level Convolutional Networks for Text Classification (github.com).

The dataset used can be found here: Yelp Review Polarity | Kaggle

Conclusion

In this project, we explored a new method to perform text classification using convolutional neural networks. With the accuracy, we were able to conclude that texts can be treated as signals too. The model outperforms current baselines such as Bag of Words and LSTMs.