# Predicting Virality on Reddit with Bidirectional LSTMs

**Kristy Duong**
Department of Computer Science
Stanford University
Stanford, CA, 94309
`kristy5@cs.stanford.edu`

**Henry Lin**
Department of Computer Science
Stanford University
Stanford, CA, 94309
`henryln1@stanford.edu`

## Abstract

The power of RNNs in the field of deep learning in recent years has led to great strides in NLP with breakthroughs in machine translation, question-answering, and many other fields. In this paper, we explore the use of bidirectional LSTMs to predict virality of submissions on Reddit, a widely used discussion forum on the Internet. We discovered that due to imbalances in the dataset and the presence of multiple, extreme outliers (i.e. viral posts), a classification approach fared much better than any regression-based model. We ultimately obtained a final dev accuracy of 31.0% and a test accuracy of 30.5%.

## 1 Introduction

As social media posts continue to vie for audience attention, it has become of increasing interest to content creators *how* to create posts that perform well and to social science researchers *why* certain posts perform well. One of the most popular sites today is Reddit, self-dubbed the "front page of the Internet" and home of over millions of users. Popular posts will become up-voted within a matter of hours and then fade into the background as new content becomes available. Previous work on Reddit has analyzed the dynamics of the Reddit community as well as the performance of text and image posts on the platform. Given the variety of communities on Reddit and Reddit's large user base, it is a treasure trove of information, including that related to virality, user sentiment, networks, and numerous other areas.

With the amount of data and computational power available, models like neural networks have become widely popular and successful across a number of fields like computer vision and natural language processing. Tasks like facial recognition, machine translation, and neural style transfer are only some of the areas revolutionized by deep learning. In this paper, we explore the use of neural networks in predicting post virality on the internet. We experimented with several different architectures before settling on a bidirectional LSTM model with cross-entropy loss. We tried different NN layers, loss functions, and hyperparameter values.

## 2 Related Work

Previous work in this field has primarily been done to analyze how to target social media content to specific audiences. Lakkaraju [5] et al. looked at temporal, language, and community features on post resubmissions to track a post's performance over time. Though not directly related, there has also been much network analysis research in this area, which attempt to analyze and predict network interaction. [1] [2]

## 3 Approach

Given the temporal nature of language, we decided on a RNN model would be the best approach for our needs. We ultimately settled on a bidirectional LSTM [3] RNN model followed with a fully connected layer that processes the final hidden state which is then used for the loss calculation. We also incorporated an embedding layer to properly convert input words into GloVe vectors, allowing us to utilize pre-trained word embeddings from the Wikipedia corpus. [6] We experimented with the output of the fully connected layer and the cost functions to determine what approach would be most effective as described below. A visual representation of our model can be found at Figure 1. To train our model, we used minibatch gradient descent with an Adam Optimizer [4] and performed hyperparameter tuning.

## 4 Experiments

### 4.1 Data Pre-processing

The raw data we worked with originally came from `https://files.pushshift.io/reddit/submissions/`, a publicly available repository of Reddit data organized into compressed JSON files timestamped by month. Given the size of Reddit, we limited our dataset to all submissions to the community r/AskReddit from September 2018. This yielded 235,609 data points. Each data point consisted of a title and a score (determined by individual users up-voting or down-voting the post).

To convert the title from words into an input suitable for a deep learning model, we also made use of publicly available pre-trained GloVe word embeddings to map each word to a multidimensional number encoding [6]. More specifically, we used 50-dimensional word embeddings to encode each title before passing it to the LSTM layer.

### 4.2 Data Augmentation

Some initial analysis of the score distribution for our data highlighted some very drastic imbalances in the dataset. For example, across the 160,000 examples we used as our training set, less than 1000 examples had a score of greater than 1000, and well over half of the training set consisted of labels of either 0 or 1. We tackled this issue using several techniques to help mitigate the bias present in the dataset. We used data augmentation to create copies of the rare examples (score > 1000) so that the model would be able to view them enough to learn their distinct characteristics. Additionally, we set the max score to be 1000 as the outliers with scores of tens of thousands or even higher were causing a huge impact in our model training. Because there are so few data points in that range, we assumed any score that high was equivalent to virality.

### 4.3 Loss Functions

#### 4.3.1 Mean Squared Error Loss

As indicated in the name, mean squared error takes the squared loss of the errors and averages them to compute the loss. This is typically the most popular loss function for regression tasks, and in almost all cases, provides a good result or a baseline upon which to improve. MSE, however, is also very vulnerable to the influence of outliers, something that resulted in this being the non-ideal loss function for our virality task. Formally, it is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

#### 4.3.2 Huber Loss

After noticing that our regression model was highly affected by outliers, or viral scores, we decided to use the Huber loss function because it is typically used in robust regression since it is less sensitive to outliers than squared error loss. The main idea of the Huber loss is to have quadratic loss at small values and change to linear loss at large values to minimize the effects of large values. The Huber

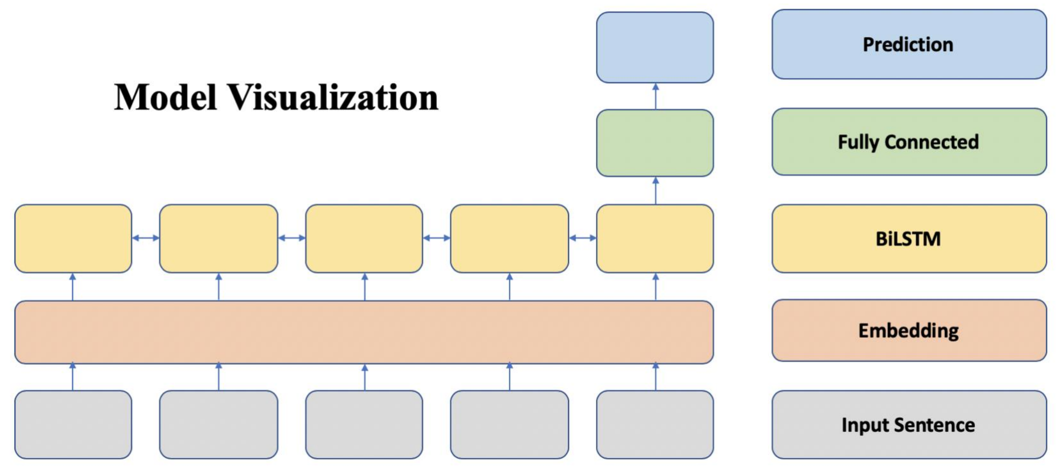

Figure 1: Model Visualization

loss function is defined as follows:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & else \end{cases}$$

#### 4.3.3 Cross Entropy Loss

Cross-entropy loss measures the performance of classification models by directly penalizing the probability in the output calculated based on how confident the prediction is. For example, predicting label 1 with probability 0.7 would incur a much lower cost than predicting label 1 with probability 0.4.

### 4.4 Regression vs. Classification

#### 4.4.1 Regression

Our regression model determined accuracy by rounding the model's prediction to the nearest whole number and comparing it against the actual score. Because our regression model had a smaller and stricter margin of error than our classification model, it may attribute to its lack in performance. This was particularly true when our regression model processed Reddit posts with higher ("viral") scores, which our model likely considered outliers, skewing the model's performance.

#### 4.4.2 Classification

Our classification model separated Reddit scores into 20 buckets with the following inclusive lower bounds: [0, 1, 2, 5, 10, 20, 30, 40, 50, 60, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]. Because most Reddit posts score below 100, we decided to use more fine-grain buckets for smaller scores, and we split 0, 1, 2 into their separate buckets because the majority of data points fall into these buckets. Additionally, we could also use this breakdown to define virality as a Reddit post with score greater than or equal to 1000.

## 5 Results

Our initial regression model using MSE Loss served as a starting point, and after training using Adam, we found that the extreme outliers that had not yet been capped at 1000 proved to be too much of a bias in the model. This meant the MSE model could not predict the small scores of 0 or 1 that constituted most of the dataset. Instead, it would output predictions around 14.8710 with some differences in decimal places.

Our next approach replaced the MSE Loss with Huber Loss, and we hypothesized that the linear loss component of Huber Loss may be able to resist the influence of large scores. Even with this approach though, we ran into the same problem, leading us to conclude that regression models would not be an effective approach to our problem.

Our best results came from our classification model using Cross Entropy loss and 20 classes to divide submission scores into buckets, discussed above. Using this approach, we managed to obtain the following accuracies for each of our datasets. Figure 2 displays the loss curve recorded from our most successful model. The minibatch gradient descent was extremely noisy, so we determined the length of training based on the decrease of training/dev loss per each epoch, not for each minibatch. As you can see, this went down each epoch steadily.

| Training | Dev | Test |
|---|---|---|
| 40.5% | 31.0% | 30.5% |

## 5.1 Hyperparameters

After deciding on a classification approach, we performed hyperparameter tuning to help obtain optimal results, and we achieved a good configuration with the following parameters:

glove_vector_dimensionality = 50
learning_rate = 0.001
hidden_size = 2048

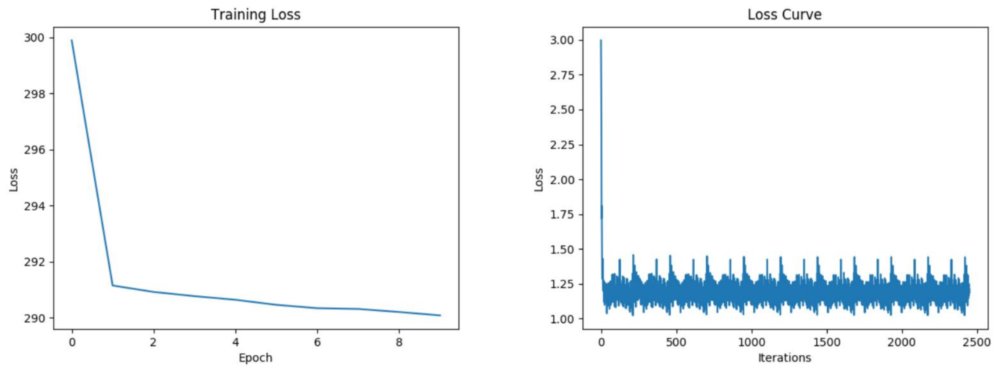

Figure 2: Training Loss per Epoch and Batch Gradient Descent

## 5.2 Error Analysis

Looking at incorrectly predicted examples in our dataset and comparing them to some correctly labeled data, it becomes clear why this is such a difficult task to perform well on. For example, the following two examples are present in the dataset:

| Question | Score |
|---|---|
| Have you ever called 911? If yes why? | 1 |
| Why did you call 911? | 1279 |

To any human individual, these questions would look incredibly similar and they do possess similar/identical words meaning the GloVe vectors would also be very similar. Despite this though, one achieved virality and the other did not. Groups of examples like these make it extremely difficult for our model to learn properly since two very similar inputs map to completely different outputs. This indicates that there may be other features that serve as better predictors or would complement our GloVe embeddings nicely, such as time of day or comment count.

Further inspection of our results showed that the vast majority of examples were classified as either "1" (score of 1) or "19" (score of 1000) or above. Compared to the true label, this indicated that the model was unable to learn any meaningful difference between the scores of 0, 1, 2, and other similarly small scores. We suspect this is because there is little or no difference at this level, and it may depend other factors such as user activity during post submission time that we did not take into account.

## 6 Conclusion and Future Work

Overall, we discovered that a classification model proved to be more effective than a regression approach despite the values given in the raw data. We also discovered that a very small percentage of posts on Reddit actually do go viral, meaning many additional data pre-processing steps must be taken to ensure that our model sees viral examples and learns from them within a reasonable number of epochs.

In our future work, we would like to continue refining this model and perhaps test alternative models to better predict viral Reddit posts. More specifically, we would like to explore the use of attention [7] in our model to provide insights about what particular words are weighted highly and may be good indicators of virality.

There are many other avenues worth exploring however, such as an ensemble model with both classification and regression components or the incorporation of metadata in the prediction. We hypothesized that the text title would be a good indicator of virality, but we also believe that using the metadata can only improve performance in future attempts. Another model we that may be interesting to explore is experimenting with a binary classification model that will flag whether a post is viral or not, since most of the labels in our dataset ended up being class "1" or "19" anyways.

## 7 Contributions

Henry Lin wrote the data cleaning/pre-processing code, monitored model training on AWS instance, created visualizations, performed hyperparameter tuning, and contributed to the poster. Kristy Duong wrote the regression and classification models, pre-processed data, and contributed to the poster.

## 8 Github Repository

Our code is located at `https://github.com/henryln1/CS230`. We based our implementation off publicly available repositories located here:

```
https://github.com/MorvanZhou/PyTorch-Tutorial/blob/master/
            tutorial-contents/403_RNN_regressor.py
https://www.jessicayung.com/lstms-for-time-series-in-pytorch/
```

## References

[1] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pages 925–936. ACM, 2014.

[2] Maria Glenski and Tim Weninger. Predicting user-interactions on reddit. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 609–612. ACM, 2017.

[3] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. What's in a name? understanding the interplay between titles, content, and communities in social media. In *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[6] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.