A Deep Learning Analytic Suite for Maximizing Twitter Impact

Zhao Chen
Department of Physics
Stanford University
Stanford CA, 94305
zchen89[at]stanford.edu

Alexander Hristov
Department of Physics
Stanford University
Stanford CA, 94305
hristov[at]stanford.edu

Darvin Yi
Department of Biomedical Informatics
Stanford University
Stanford CA, 94305
darvinyi[at]stanford.edu

Abstract

We present a series of deep learning models for predicting user engagement with twitter content, as measured by the number of retweets for a given tweet. We train models based on classic LSTM-RNN and CNN architectures, along with a more complex bi-directional LSTM-RNN with attention layer. We show that the attention RNN performs the best with 61% validation accuracy, but that all three deep learning models outperform human accuracy for the same task.

1 Introduction

Twitter has quickly become one of the leading platforms for content sharing, allowing both private citizens and public corporations and everyone in between to quickly share both personal and commercial content. Content providers seek to maximise the visibility of their posts and users benefit when they find content relevant to their interests. Therefore, it is of great interest to find what kinds of tweets are correlated with high retweet/favorite counts. Such information can inform content providers what to produce and can help filtering algorithms deciding what content to show users. Additionally, tweets are compact in length but rife with casual language, making them ideal candidate for natural language processing (NLP) techniques. We thus propose to develop an analytic suite for tweets based on deep learning NLP techniques to predict the reception of tweets based on twitter accounts and analyze how language patterns affect retweet count per account.

In order to adequately explore our model space, we propose multiple deep learning models to tackle this problem. In increasing order of complexity, these are:

1. Vanilla LSTM-RNN Model
2. Contolutional 1d Alex-Net
3. Bi-directional LSTM-RNN with an Attention Layer

We expect that, due to the complexity of factors that may lead Twitter users to a tweet, and due to the diversity amongst members of the Twitter community, that the more flexible, more complex models will have better performance on this problem. To further illustrate this point, we will compare our model with human performance at the same task, and show that the most complex models exceed human ability to predict Twitter impact.

2 Background/Related Work

Recurrent neural networks have become standard building blocks in various language-related tasks, and are widely used in machine translation [15], sentiment analysis [7], and even complex question/answer systems [8]. The LSTM in particular [6] has become widely used as the go-to building block for such RNN models due to their much more robust memories and ability to selectively forget inconsequential subsequences compared with more vanilla modules.

For language processing tasks that involve longer sentences or information that is not purely presented in chronological sequence (a central assumption to training classic RNNs), the bidirectional RNN [14] is a tempting choice. Bidirectional RNNs allow for sequences to be trained in both directions, and by concatenating hidden state vectors for both the forward and reverse legs of the BiRNN, we can produce outputs at each time step, each of which were derived by the same total number of affine/nonlinear transformations. These outputs can then be pooled to produce a final prediction.

The pooling itself can be accomplished by an attention layer, which have recently come into popular use with RNN language models [16]. When our BiRNNs produce outputs at each layer, it is likely that some outputs at certain positions will be more important in our predictive model than others. In the spirit of deep learning, we would like the neural network to be able to decide itself which outputs are more important. This is accomplished by placing an additional affine transformation and nonlinearity at each output, and dotting the result with a context vector trained in parallel, which then produces attention coefficients which inform our model how much to weight each output in the final prediction. For more details, see Section 3.

In addition, although generally not used in language models, convolutional networks have also enjoyed some measure of success in language related tasks, such as Twitter sentiment analysis [10]. That task and the task discussed here are both very similar, in both high level concept and practical details. The former uses the text to classify a tweet as either positve, neutral, negative in sentiment and, as discussed in Section 3 the latter classifies to text into high, medium and low popularity/impact. We are therefore interested in how convolutional neural networks perform in our impact prediction problem.

3 Approaches

We first describe our recurrent models before proceeding to the CNN. For our RNN models, we use LSTM cells as our basic building blocks [6]. These models are now widely favored over simple recurrent neural networks (SRNN's), which suffer from optimization problems described in [2]. The forward-pass equations for the LSTM for one step are summarized below:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$

$$\widetilde{C_t} = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3}$$

$$C_t = i_t * \widetilde{C_t} + f_t * C_{t-1} \tag{4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \tag{5}$$

$$h_t = o_t * tanh(C_t) \tag{6}$$

3.1 Classic LSTM-RNN

For a baseline model, we train a classic LSTM-RNN model, where we have a simple softmax layer

$$\hat{y} = \text{softmax}\left(W^S h_T + b^S\right) \tag{7}$$

applied to the output of the last hidden layer.

A schematic of this model is shown in Figure 1. Though variants of LSTM cell have been published in the literature, extensive studies have shown that modification of this cell does not appreciably improve the performance of the model on many tasks [4].

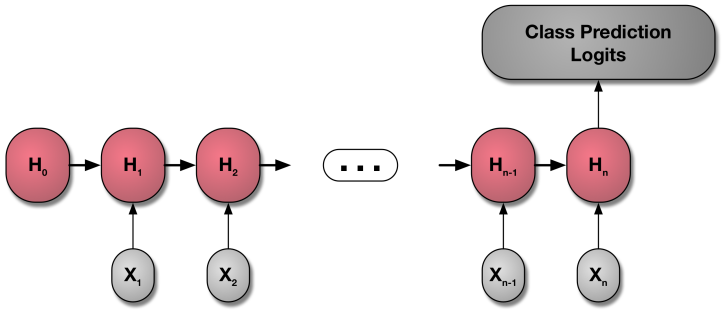

Figure 1: **RNN classifier** Classical RNN classifier using LSTM cells.

3.2 Bi-Directional Attention Model

To improve on the above model, we make two key changes. First, we do not extract a prediction from the end of the text alone, so we replace the RNN with a bidirectional RNN. To extract a prediction from this, we pool output from each set of cells. This bidirectionality has previously been shown to improve classification in local text tasks, such as phoneme classification [3] Second, we suppose that not all words contribute equally to the representation of the text meaning. Therefore, we introduce an attention mechanism which extracts the most important words and aggregate the representation of those informative words to form a prediction of tweet usage. The bidirectional attention model used here is based on the model described in [16]. Given outputs $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ produced at each time step i, we first form the concatenated output vector $[\overleftarrow{h}_i, \overrightarrow{h}_i]$. We then create three new trainable parameters, W_w, b_w, and u_w, such that we calculate

$$v_i = \tanh(W_w[\overleftarrow{h}_i, \overrightarrow{h}_i] + b_w) \tag{8}$$

We then compare the vector v_i with a vector u_w, and the strength of this comparison tells us how much attention to pay to the word v_i. More precisely,

$$\alpha_i = \frac{\exp(v_i^T u_w)}{\sum_j \exp(v_j^T u_w)} \tag{9}$$

where the denominator is summed over all time-steps. This produces a probability distribution in the α_i over all steps, and we then take a weighted sum over this distribution:

$$\hat{y}_i = W^S[\overleftarrow{h}_i, \overrightarrow{h}_i] + b^S \tag{10}$$

$$\hat{y} = \sum_{i=1}^{45} \alpha_i \hat{y}_i \tag{11}$$

The prediction is then taken as the argmax of the final class scores $\hat{y}$. We can also take the softma scores softmax$(\hat{y})$ and take the expected value over the resultant distribution corresponding to classes $\{0, \ldots, C\}$ to arrive at an expected class

$$E[C] = \sum_{i=1}^{C} i\hat{y}[i] \tag{12}$$

This is actually meaningful in our context as our classes represent a natural ordering for tweet impact. The schematic of this model is shown in Figure 2

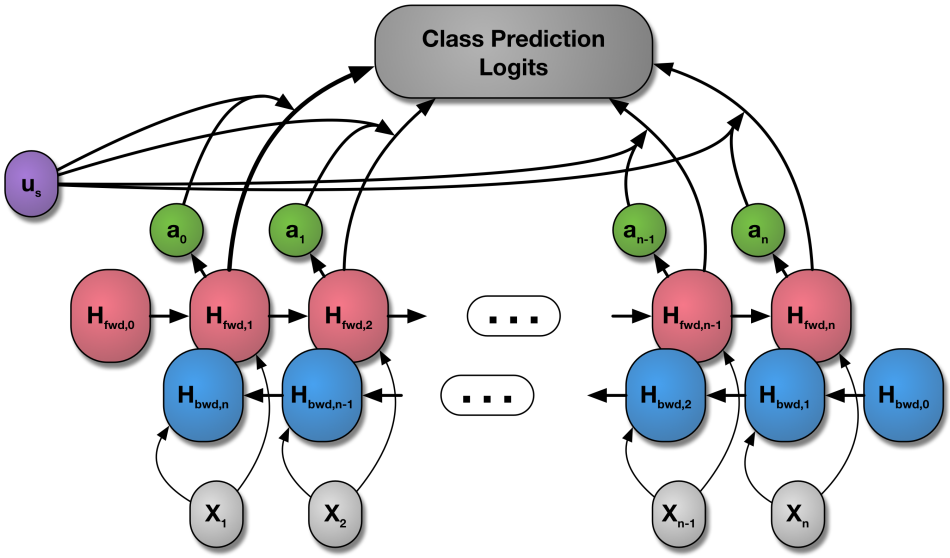

Figure 2: **RNN classifier with attention** A bidirectional RNN with outputs at each time-step further weighted by attention weights α_i.

3.3 Convolutional 1d AlexNet

Due to the success of convolutional neural networks in some language tasks, we choose to explore whether a classic CNN architecture can perform well on our impact analysis problem. For simplicity, we use a modified version of AlexNet, a famous CNN architecture which came into prominence after winning the 2012 iteration of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12]. Our model has similar patterns of conv layers, stride, and filter sizes, but we remove most pooling layers due to the smaller sizes of our inputs. Of course, because our sequences lie in one dimension, we also collapse our filters to be one dimensional. Our CNN architecture is as follow:

1. 96 11×1 convolutional filters, stride 1.
2. 256 11×1 convolutional filters, stride 1.
3. 384 11×1 convolutional fliters, stride 1.
4. 384 11×1 convolutional filters, stride 1.
5. 256 3×1 convolutional filters, stride 1.
6. 45×1 maxpool.
7. 256 FC.
8. 256 FC.
9. 3 FC (affine only).

The maxpooling collapses all information across the 45 time steps that we begin with in our model, which we hope will mitigate the effects of the period padding we used to preprocess our data (see next Section). A schematic of this model is shown in Figure 3

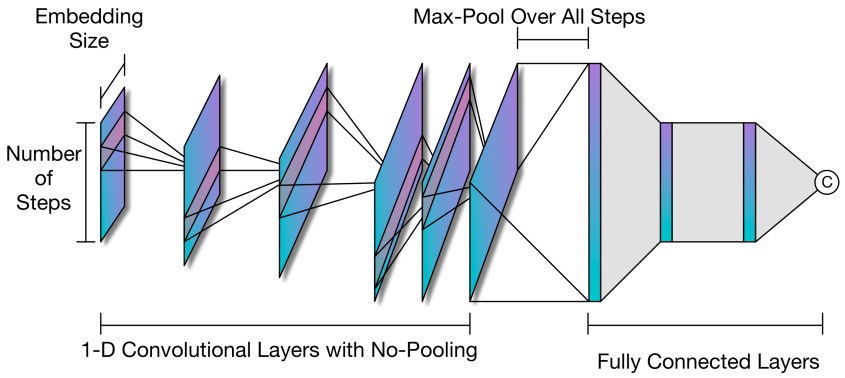

Figure 3: **CNN classifier** A one-dimensional CNN architecture based loosely on AlexNet for our impact classification problem.

4 Experiment

4.1 Data

Twitter content from thirty news sources were compiled dating back nearly two years, depending on source [1]. A total of three thousand posts were collected for each news source. Each posted item has text and impact metrics (retweets and favorites).

The impact metrics of each tweet are preprocessed per account to classify the post low impact, medium impact, and high impact. This classification is done by partitioning the data into ordered thirds based on the number of favorites, but alternative methods can easily be integrated into our treatment. Then, all data for all accounts are pooled to form our final data set (see Figure 4). We thus pose our problem as a 3-class classification problem. The classes are created on a per-account basis as this allows us to normalize for the fact that some accounts have higher mean impact than ohter accounts. We process Approximately 15% of the data is set aside as a validation set.

All tweet text is first pre-processed by using 50-d GloVe word vectors [13] obtained from nlp.stanford.edu to turn length each tweet into an array of first dimension 50. Words were padded with periods to standardize the length of each tweet to 45 words (the max length of any tweet in our data set), giving each tweet a representation in $\mathbb{R}^{50} \times \mathbb{R}^{45}$. This allows us to have standardized, same-length inputs.

4.2 Prediction Accuracy

A human performance baseline was used in which an author (A.T.H.) trained on a set of 150 tweets, and then evaluated on a further set of 250 tweets. Though this set is considerably smaller, we feel it is a fair representation: first, the binomial variance estimated in this way is quite small and second, humans are much slower at this task so the time to train a human was within an order of magnitude to the time to train some of the simpler models included here.

[1] The sources came from the following news organizations: BBC, CNN, USA Today, The Harvard Business Review, Gizmodo, Fox News, ESPN News, E! News, New Scientist, The Economist, The Wall Street Journal, Reuters, Newsweek, The Huffington Post, The New York Times, Time, Rolling Stone, Slate, Mashable, and Tech Crunch. In some cases, multiple twitter accounts were used per source, so the total of thirty twitter feeds reflects fewer sources.

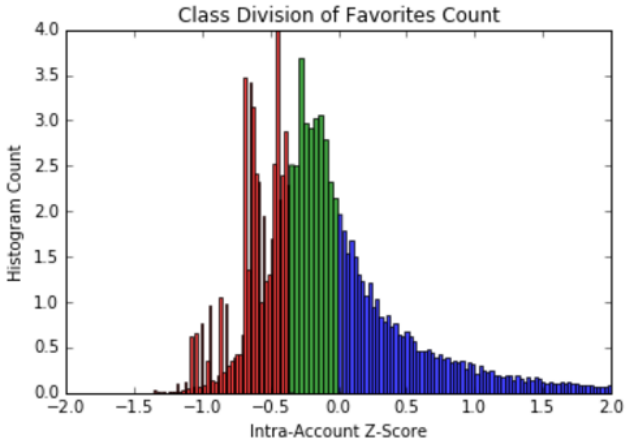

Figure 4: **Class distribution** Tweets separated into three classes, based on their z-score (calculated separately for each account before all data is pooled).

Models are run on AWS Grid K520 GPUs, and code was written in TensorFlow [1]. After running our models, we report the validation accuracies as shown in Table 1 after an early stopping condition in model training was reached (generally when the validation accuracy of the model ceases to improve for two epochs). Training time varied for the different models, but generally ranged between 2 and 10 hours.

Basic hyperparameter search was performed. Due to time limitations, the number of hyperparameter configurations tested for each model type varied from half a dozen to two dozen. All models were also trained with batch size of 128 and the adam update rule for minimizing cross entropy loss [11]. We can see that the attention model performs the best, achieving training and validation accuracies

Table 1: **Performance Comparison.**

Algorithm	Description	Training Acc.	Validation Acc.
Human Baseline	A human tried to guess how popular a tweet would be.	-	0.39
Basic RNN	This was a vanilla RNN. We used code from the second problem set.	0.75	0.51
Vanilla LSTM	This is similar to the Basic RNN model above, but we used a LSTM cell instead of a simple affine transformation.	0.78	0.59
1-D CNN	A 1-dimensional convolutional neural network based on the architecture of AlexNet.	0.76	0.52
Attention Model	A Bi-Directional RNN with an attention layer placed on top.	0.82	0.61

of 82% and 61% respectively. The basic RNN (simple tanh nonlinearities after an affine transformation of nte inputs) performs the worst, beating the human baseline model but only achieving 51% validation accuracy. The 1-D CNN and Vanilla LSTM models fall in the middle.

At test time, the attention model also provides reasonable results. For example, refer to the results in Table 2. Even for example tweets that are similar in structure, the RNN can detect changes in descriptive words embedded deep in the middle of a sentence and adjust the expect impact accordingly.

Table 2: **Test-time behavior of attention model**

Tweet	Predicted Impact (E[C])
stock market suffers massive losses amid nationwide strikes	1.85
stock market suffers minor losses amid nationwide strikes	1.00
stock market stable	0.58

It is instructive to see how the model trains over time, and we show this in Figure 5. We can see that there is some measure of overfitting, which suggests that we could increase the regularization penalty to compensate. However, generally, increasing the regularization penalty did not lead to signifiant improvements in our case, and we suspect that a more sizable improvement can be achieved by deepening the model (i.e. adding more layers atop the first layer outputs of our RNN models). More of this is discussed in Section 5.

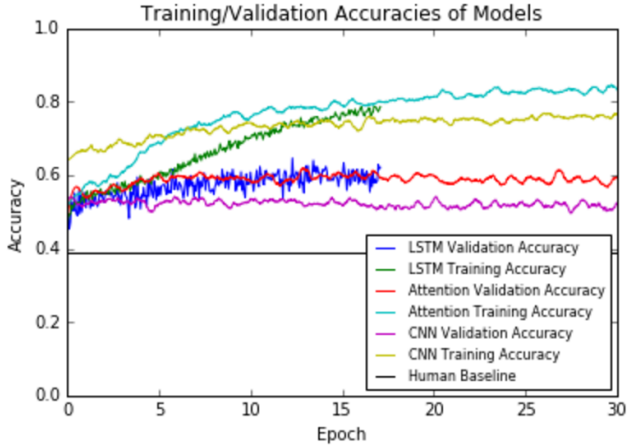

Figure 5: **RNN classifier** Classical RNN classifier using LSTM cells.

It is, however, interesting to note that all our models outperform human performance at the impact classification task. This suggests that although 61% accuracy may not seem high, impact classification is a much more complex problem than tasks like sentiment analyses where humans tend to do fairly well. We are thus asking our models to look for correlations between impact and Twitter language that are not quite clear even to humans, making achieving 60% accuracy meaningful.

5 Conclusion

Deep learning provides improvements upon human accuracies in the Twitter impact classification task. In terms of different deep learning models, the bidirectional RNN with attention layer performed the best, possibly because it (1) incorporates information over the entire tweet with a flexible weight, something a classical RNN cannot do, and (2) still explicitly takes sequence position into account, something a CNN cannot do.

Given the improved performance of the algorithm over a human classifier, we find it highly intiguing to speculate on the use of this system as a content curator. The quantitative model purely reacting to user engagement seems freer of the biases of a human editor, who might naturally conflate "what I want" with "what others want" in a common case of substituting an easy question for a more difficult one [9]. However, a more fair comparison might come from the "wisdom of crowds" in which this bias can be reduced by averaging predictions over many individuals.

Another possible direction would be to explore deeper architectures to introduce even more nonlinearity into our model, and to perform more thorough hyperparameters search to combat overfitting. The former may improve performance of the model by quite a lot; the weak human baseline for this problem suggests that the problem itself is highly complex and may need a deeper rather than a wider model.

Another useful direction to take these results might be to include user-specific input into a more dynamic personalized recommender system, which remains an area of ongoing work for item recommendation [5], but which has yet to be investigated in the NLP context.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[3] Alex Graves and Jrgen Schmidhuber. Framewise phoneme classification with bidirectional {LSTM} and other neural network architectures. *Neural Networks*, 18(56):602 – 610, 2005. {IJCNN} 2005.

[4] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728, 2014.

[8] Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014.

[9] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[10] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

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

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[14] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

[15] Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. Translation modeling with bidirectional recurrent neural networks. *EMNLP*, 2014.

[16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification.