CS230

Predicting Political Affiliation from Tweets

Jorge Cordero
Dept. of Electrical Engineering
Stanford University
icoen@stanford.edu

Eddie Sun
Dept. of Mechanical Engineering
Stanford University
eddiesun@stanford.edu

Zoey Zhou
Dept. of Electrical Engineering
Stanford University
cuizy@stanford.edu

Abstract

In this work, we classify political affiliation from tweets using a convolutional neural network (CNN) and a long-short-term-memory network with attention (LSTM-Attn). The balanced dataset that we use to train the models on contains 86,460 labeled tweets from Republican and Democrat congressmen. After hyperparameter tuning on both models, the CNN model achieves a test set classification accuracy of 82.4% and the LSTM-Attn. model achieves an accuracy of 84.1%. We also experiment with gated recurrent units (GRU) and bidirectional variants of both LSTM's and GRU's and report the performance of each. We find that the LSTM-Attn. model (the higher performing model) correctly classifies the tweets that politically-informed humans could also correctly classify, but struggles to classify short tweets, tweets with multiple people mentioned, and tweets with no political content. (Github Link: https://github.com/icoen/CS230P)

1 Introduction

The knowledge of public opinion is an extremely powerful tool. Companies with public opinion knowledge can use it to decide what products to create and how to best market them. Governments can use public opinion knowledge, such as political affiliation, approval level, and general interests, to determine how to best serve their constituents. In today's political climate especially, there appears to be a large disconnect between politicians and citizens. Improved opinion surveying could remedy this disconnect.

Currently, opinion polls are used to gauge public opinion, which only sample a small portion of a population. However, it is difficult to determine the opinions of large populations, such as that of an entire state. Opinion polls are carefully designed such that they try to sample a representative sample of the entire population and that they report confidence levels and other statistical measures, but extrapolation to the whole population is still an inherent flaw.

For this project, we use Deep Learning to build a framework that may survey the opinions of a large population. Specifically, we designed a political orientation classifier that takes a Twitter tweet as an input and classifies it as being Republican or Democrat based on the contents of the tweet. We implemented two different models to accomplish this task: a convolutional network (CNN), and a long-short-term-memory (LSTM) network with attention. Each model and its respective performance is described in more detail in the following sections.

2 Related work

In recent years, there have been many published works on political party classification using both Deep Learning and non-Deep Learning methods. Bakliwal et al. (2013) performed sentiment classification of political tweets using support-vector-machines (SVM) and achieved 61.5% accuracy in a 5-class sentiment classification test [1]. However, the tweet labels were labeled by "political experts" and are thus subjective. Ding et al. (2016) performed both sentiment classification and political party classification using Naive Bayes, logistic regression, SVM, and decision tree classifiers on a small training and development dataset. The Naive Bayes model achieved approximately 77% accuracy on sentiment analysis using 187 handpicked tweets while the SVM achieved 82% accuracy on political party classification using 117 handpicked tweets during the three presidential debates[2]. Biessmann et al. used a multinomial (k political parties, in Germany) logistic regression to classify political affiliation, achieving around 76% accuracy on Facebook posts [3]. While these statistical machine learning models perform rather well, are easy to implement and train, and can be trained on less data, deep learning methods achieve higher accuracy.

Deep Learning approaches to this problem are less common, but the best models perform better. Iyyer et al. (2014) used a recursive neural network (RNN) model to classify political ideology from sentences and achieved around 70% accuracy for a 3-class classification problem [4]. Rao and Spasojevic (2016) improved on this and used an LSTM model to classify social media posts as Democrat vs. Republican, achieving 87.5% accuracy with 336,000 training tweets from 27,130 twitter users with known political leaning identified by twitter lists [5]. Well-trained neural networks tend to perform better because of CNN and RNN models' ability to process word order and grammar, as opposed to a bag-of-words model used by non-neural network models.

From the literature review, we decided to implement CNN and RNN models to accomplish the political party classification task, since they are the current state-of-the-art for sequence based learning tasks. In addition recent papers have shown that adding attention can also improve the performance in text classification tasks [21], [22], [23]. A network architecture literature search was conducted specifically on CNN and RNN models relating to tweet classification. While few CNN's and RNN's have been reported specifically concerning tweet-based political party classification, there are many previous works on tweet sentiment classification, which is a related task. We therefore consider the architectures of previous tweet sentiment classification networks. There are two main architectures: 1) a two layer model consisting of a single LSTM layer [5-8] followed by a fully-connected (FC) layer and 2) a two layer model consisting of a convolutional/max-pool layer [9] followed by a fully-connected (FC) layer. We based our political tweet classifier on these two models while also adding an attention layer as well as experimenting with other recurrent cell types (GRU, bidirectional).

3 Dataset and Text Preprocessing

The dataset we are using for this project contains 86,460 labeled tweets, the most recent 200 tweets by 211 Democrat and 222 Republican congressmen collected in May 2018 [10]. Approximately 49% of the tweets are from Democrats and 51% are from Republicans, which were proportional to the fraction of seats currently held by each respective party in Congress before the 2018 midterm elections. We split the dataset into 80/10/10 between train/dev/test sets. Unfortunately, tweets were truncated to 141 characters, so we used an html parser [11] to download the full tweet from html links included for original tweets (we could not download the full tweet for retweets). We perform the standard text preprocessing of tokenization, pad each tweet to maximum word length of the training data (65 words) and convert to lower case. We first tried feeding in the raw tweet, we subsequently tried these in turn: removed hyperlinks, removed hashtags, removed mentions, removed punctuation, kept quotations but removed other punctuation, and separated the hashtags and mentions at capital letters. We used the combination that produced the highest development accuracy: removed hyperlinks, removed punctuation, and separated hashtags at the capital letters. We also tried using the GloVe twitter embedding (50 dimensional and 200 dimensional vectors) but found that accuracy improved when using a trainable embedding layer [12].

4 Methods

Based on previously published related work, we use two models for our task: a 1-D CNN model and a LSTM-Attention model. We use the CNN as a baseline model since it has been previously attempted. We chose to create an LSTM-Attention model since this type of model has not previously been used for this task and may yield better performance than previous models (LSTM alone was used previously in [5]). We also experiment with gated-recurrent-units (GRU) and bidirectional LSTM's and GRU's. All models are coded using Keras [13] with Tensorflow backend [14]. Since our dataset is balanced between positive and negative examples, we use accuracy as the main evaluation metric. We also consider the macro precision, recall (averaged over each of precision and recall of both categories) and F1 from those macro scores. We can use macro scores as a metric because we have a balanced dataset. Cross-entropy loss is used as the loss function, where y is the label and $\hat{y}$ is the prediction.

$$\mathcal{L}(y, \hat{y}) = y \log(\hat{y}) - (1 - y)\text{log}(1 - \hat{y}) \tag{1}$$

4.1 1-D Convolutional Neural Network (CNN)

The theory and model architecture for the 1-D CNN used for this task is based on Ref. [9], with reference code originally from [15]. After the input tweet is embedded, the model consists of a 1-D convolutional layer followed by a max-pool layer. For each convolution in the convolutional layer, the filters convolve over the nearest w words, where w is the window size. The max-pool layer pools over the outputs of each filter. The output of the max-pool is then fed to a fully connected layer before being fed to a single sigmoid output neuron. We use dropout [16] and L2 regularization in the fully connected layer to regularize.

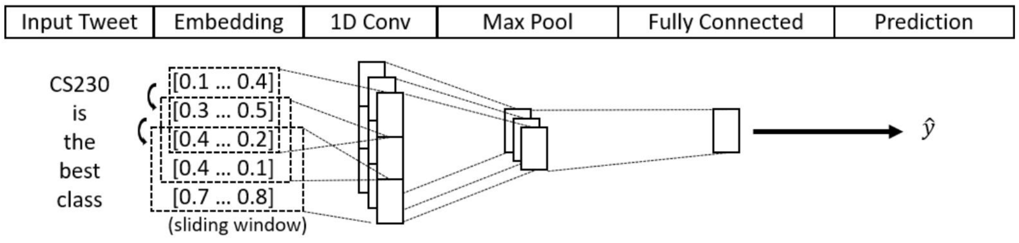


Figure 1: CNN Model Architecture

4.2 Long Short Term Memory (LSTM)

LSTM neural networks are superior to traditional RNN's in that LSTM's do not suffer from vanishing gradient problems and can learn long-term dependencies (i.e. relationships between the first and last word of a sentence). An example of a standard LSTM network is shown in Figure 2.

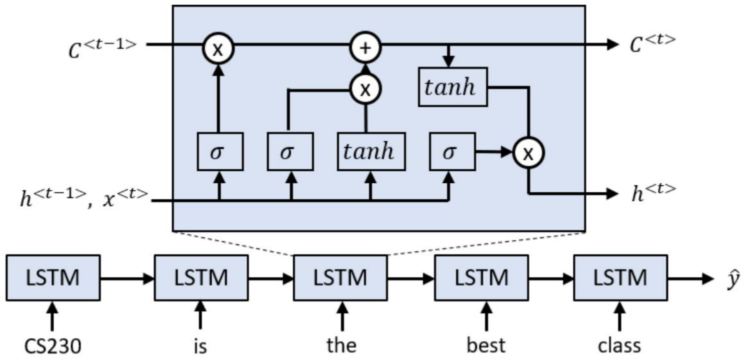


Figure 2: Example LSTM model architecture, with the detailed view of a single LSTM cell.

In the above figure, C is the cell state, t is the timestep or the t^{th} word in the sequence, h is the cell output, x is the input (embedded word), σ and $tanh$ are the sigmoid and arctangent functions respectively, and the x and $+$ operators enclosed in a circle are multiplication and addition operators. The LSTM cell in the figure represents these equations:

$$C^{<t>} = \Gamma_u \tanh(W_c[h^{<t-1>}, x^{<t>}] + b_c) + \Gamma_f \, C^{<t-1>} \tag{2}$$

$$h^{<t>} = \Gamma_o \tanh C^{<t>} \tag{3}$$

$$\Gamma_u = \sigma(W_u[h^{<t-1>}, x^{<t>}] + b_u) \tag{4}$$

$$\Gamma_f = \sigma(W_f[h^{<t-1>}, x^{<t>}] + b_f) \tag{5}$$

$$\Gamma_o = \sigma(W_o[h^{<t-1>}, x^{<t>}] + b_o) \tag{6}$$

where W and b are weights, and Γ_u, Γ_f, and Γ_o are the update, forget, and output gates. Using these gates, the LSTM can selectively filter for only important filters to update it's cell state C and thus can learn long term dependencies.

We also experimented with gated-recurrent units (GRU) with a similar architecture replacing LSTM with GRU and the fully connected layer with a max pooling layer. We tried bidirectional LSTM's and GRU's (since it only requires changing one line of code in Keras). Due to space constraints, we do not describe these models in detail.

4.3 Attention

For long sentences, an Attention model attached to an LSTM network can provide improved performance. Attention allows the network to learn how much "attention" to pay to each word in the sequence by assigning weights $\alpha^{<0,t'>} = exp(e^{<0,t'>})/\sum_{t'=1}^{T_x} exp(e^{<0,t'>})$ and multiplying them to the output of the corresponding LSTM cell $\alpha^{<t'>}$, the result is summed. T_x is the length of the LSTM cells. Each $e^{<0,t'>}$ is learned from a small NN with input $\alpha^{<t'>}$.

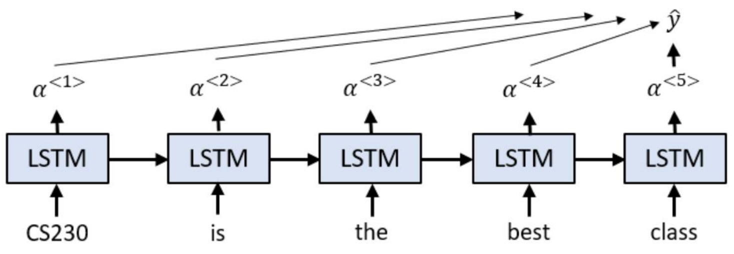


Figure 3: Example LSTM with attention model architecture.

Our LSTM-Attention model has the following architecture, which is the same LSTM model as found in Ref. [5], except with added Attention.

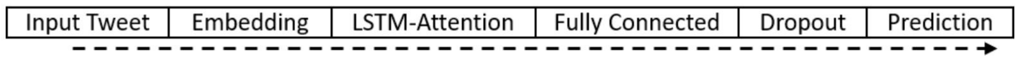


Figure 4: LSTM-Attention Model Architecture

5 Experiments, Results, and Discussion

5.1 Hyperparameter Search

From an initial test with all of the different architectures we found CNN and LSTM-Attention to have the best dev. accuracy. We choose these two architectures to pursue and optimized each using two rounds of hyperparameter searching. For both models, we first performed a broad random hyperparameter search running 100 models at once (caviar method). From the results, we then picked the combination of hyperparameters that yielded the best accuracy and then performed a fine hyperparameter search of 20 points around that combination. The hyperparameters tested and the range of each for the 1st optimization are summarized in the table below.

Table 1: 1st Hyperparameter Search

CNN Hyp-param.	Range	LSTM-Attn. Hyp-param	Range
Learning Rate α	$(10^{-4}, 10^{-2})$	Learning Rate α	$(10^{-4}, 10^{-2})$
Embedding Dim.	[50, 100, 200, 300]	Embedding Dim.	[50, 100, 200, 300]
Dropout Prob.	(0, 1)	Dropout Prob.	(0.2, 0.8)
# Filters	[32, 64, 128]	Batch Size	[32, 64, 128, 256]
L2 Regularization λ	(0, 1)	# LSTM Cells	[32, 64, 128, 256]
		# Neurons (FC)	[64, 128, 256, 512]

The following table shows the optimal hyperparameters for each model.

Table 2: Optimum Hyperparameters

CNN Hyperparameter	Value	LSTM-Attn. Hyperparameter	Value
Learning Rate	$1.0 * 10^{-3}$	Learning Rate	$3.7 * 10^{-3}$
Embedding Dim.	100	Embedding Dim.	300
Dropout Prob.	0.52	Dropout Prob.	0.44
# Filters	128	# LSTM cells	64
L2 Lambda	0.35	Batch Size	128
		# Neurons (FC)	64

5.2 Model Performance

The table below summarizes the performance of each model. The best perforamces are in bold. We also include the performance of the LSTM (no attention), GRU, bidirectional LSTM, and bidirectional GRU for reference even though we did not perform hyperparameter searches on them.

Table 3: Model Performance

Model	Train Acc.(%)	Dev. Acc. (%)	Test Acc. (%)	Precision (%)	Recall (%)	F1 (%)
CNN	**96.9**	84.5	82.4	82.4	82.4	82.4
LSTM-Attn.	94.4	**85.4**	**84.1**	**84.2**	**84.1**	**84.1**
LSTM	91.3	84.6	83.3	83.3	83.2	83.3
GRU	88.6	84.8	83.6	83.6	83.6	83.6
LSTM (Bi-Dir)	90.0	84.6	83.3	83.4	83.2	83.3
GRU (Bi-Dir)	90.6	84.4	83.6	83.7	83.5	83.6

The LSTM-Attn. model performs the best, and all recurrent models perform better than the CNN. This is because the CNN's ability to learn long-term dependencies is limited by its filter size, which can only scan over a few words at a time. In contrast, LSTM's are designed to learn these long-term dependencies. Despite performing slightly worse, the CNN still achieves remarkable accuracy given its simple model architecture.

We also note that the LSTM-Attn. model, even after hyperparameter tuning, only performs marginally better than LSTM's and GRU's without attention. This is reasonable since attention greatly improves performance in long sentences, whereas tweets are relatively short.

In addition, our accuracy is better than Ding et al's SVM method by 2.1% [2]. Furthermore their dataset was collected during the presidential debates so that there was a high uniformity of topics and sentiments. Rao and Spasojevic's LSTM model achieved dev. accuracy of 87.5% which outperforms our model [5]. Their dataset were full tweets (including retweets), about 5 times the size of ours and they did not have a separate test set. In addition we used their architecture and hyperparameters for our LSTM model which did not perform as well as the LSTM-Attn. model.

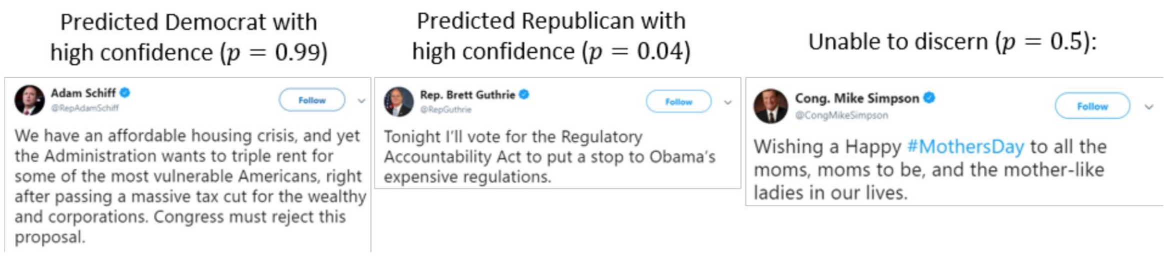
5.3 Error Analysis

The confusion matrix for the LSTM-Attention model, is shown below.

Table 4: LSTM-Attention Model Confusion Matrix

	Predicted Democrat	Predicted Republican
True Democrat	3442 (39.9%)	787 (9.1%)
True Republican	608 (7.0%)	3808 (44.0%)

The model appears to be slightly biased towards Republican predictions. We performed an error analysis on the LSTM-Attn. model, but unfortunately we were not able to determine any obvious reasons for the bias. We did note that the model performs well (classifies correctly with high confidence) on tweets that politically-aware humans could correctly classify and performs poorly on tweets that would be difficult for any human to classify, such as tweets with no political content, short tweets (<6 words), and tweets with multiple people mentioned in the tweet.


Predicted Democrat with high confidence ($p = 0.99$)
Predicted Republican with high confidence ($p = 0.04$)
Unable to discern ($p = 0.5$):
Follow
Follow
Follow

Adam Schiff @RepAdamSchiff

We have an affordable housing crisis, and yet the Administration wants to triple rent for some of the most vulnerable Americans, right after passing a massive tax cut for the wealthy and corporations. Congress must reject this proposal.

Rep. Brett Guthrie @RepGuthrie

Tonight I'll vote for the Regulatory Accountability Act to put a stop to Obama's expensive regulations.

Cong. Mike Simpson @CongMikeSimpson

Wishing a Happy #MothersDay to all the moms, moms to be, and the mother-like ladies in our lives.

Figure 5: Examples tweets with model confidence level

6 Conclusion/Future Work

In this project, we created and optimized a CNN and LSTM-Attention model to determine political party affiliation from tweets. These models are competitive with state-of-the-art models and achieved an accuracy of 82.4% and 84.1% respectively. The LSTM-Attention performs better due to its ability to learn long-term dependencies. However, given the CNN model's simplicity, its slightly worse accuracy is still remarkable.

Possible future work involves collecting more full tweets, using character-level embeddings, further neural network architectures searching and experimentation, hyperparameter tuning and tweet preprocessing. Vosoughi et al. developed Tweet2Vec [17], a character level embedding method specifically designed for tweets, which may perform better with tweets than GloVe. Other network architectures to experiment with include recurrent-convolutional NN's [18, 19] and bi-directional RNN's with temporal max-pooling [20].

7 Contributions

JC: AWS setup, CNN model, CNN and LSTM hyperparameter search.

ES: LSTM-Attention model, poster, final report.

ZZ: Tweet preprocessing, LSTM/GRU models, LSTM-Attention model hyperparameter search.

References

[1] Bakliwal, Akshat, et al. "Sentiment analysis of political tweets: Towards an accurate classifier." Association for Computational Linguistics, 2013.

[2] Ding, Tianyu, et al. "Sentiment Analysis and Political Party Classification in 2016 US President Debates in Twitter." (2016).

[3] Biessmann, Felix, et al. "Predicting political party affiliation from text." (2018).

[4] Iyyer, Mohit, et al. "Political ideology detection using recursive neural networks." Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. 2014.

[5] Rao, Adithya, and Nemanja Spasojevic. "Actionable and political text classification using word embeddings and lstm." arXiv preprint arXiv:1607.02501 (2016).

[6] Jiang, Keyuan, et al. "Identifying tweets of personal health experience through word embedding and LSTM neural network." BMC bioinformatics 19.8 (2018): 210.

[7] Sosa, Pedro. “Twitter Sentiment Analysis Using Combined LSTM-CNN Models.” B-Sides, 2018, konukoii.com/blog/2018/02/19/twitter-sentiment-analysis-using-combined-lstm-cnn-models/.

[8] Yuan, Ye, and You Zhou. "Twitter sentiment analysis with recursive neural networks." CS224D Course Projects (2015).

[9] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[10] Pastor, Kyle. “Democrat Vs. Republican Tweets.” Kaggle: 27 May 2018.

[11] Richardson, Leonard. "Beautiful soup." (2013).

[12] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[13] Chollet, François. "Keras." (2015).

[14] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. 2016.

[15] Britz, Denny. “Convolutional Neural Network for Text Classification in Tensorflow.” GitHub, 21 July 2018, github.com/dennybritz/cnn-text-classification-tf.

[16] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[17] Vosoughi, Soroush, Prashanth Vijayaraghavan, and Deb Roy. "Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder." ACM, 2016.

[18] Elbayad, Maha, Laurent Besacier, and Jakob Verbeek. "Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction." arXiv preprint arXiv:1808.03867 (2018).

[19] Zhou, Peng, et al. "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling." arXiv preprint arXiv:1611.06639 (2016).

[20] Lai, Siwei, et al. "Recurrent Convolutional Neural Networks for Text Classification." AAAI. Vol. 333. 2015.

[21] Yang, Yang, et al. "Hierarchical Attention Networks for Document Classification." Proceedings of NAACL-HLT, 2016. [22] Zhou, Shi, et al. "Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.

[23] Du, Huang. "Text Classification Research with Attention-based Recurrent Neural Networks." International Journal of Computers Communications & Control, 2018.