Crafting Adversarial Attacks on Recurrent Neural Networks (RNNs)

Mark Anderson, Andrew Bartolo, Pulkit Tandon
{mark01, bartolo, tpulkit}@stanford.edu

Summary

• RNNs are used in a variety of applications to recognize and predict sequential data. However, they are vulnerable to adversaries; e.g., a cleverly-placed word may change the predicted sentiment of a movie review from positive to negative.
• We built Naïve Bayes, SVM, and LSTM models to predict movie review sentiment and built two black-box adversaries. We show that NB and SVM are sensitive to these attacks while LSTMs are relatively robust.
• Finally, we implemented a recent Jacobian-based technique for generating adversaries for LSTM, and found that LSTM performance falls below 40% by replacing an average of 8.7 words. We also found examples where the classification error was brought on by a seemingly-random word, indicating that the LSTM might not be truly learning sentiment.

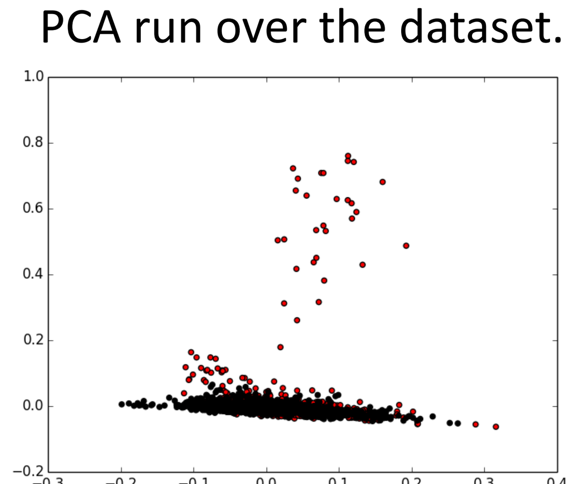
Data & Features

We train on a pre-labeled set of 12,500 positive and 12,500 negative movie reviews, collected from IMDb [1]. Reviews averaged 233 words. For compatibility with the NumPy and TensorFlow input models, SVM and LSTM reviews are capped at 250 words. We strip all punctuation from the reviews, but leave stop words.

IMDb Movie Review Dataset

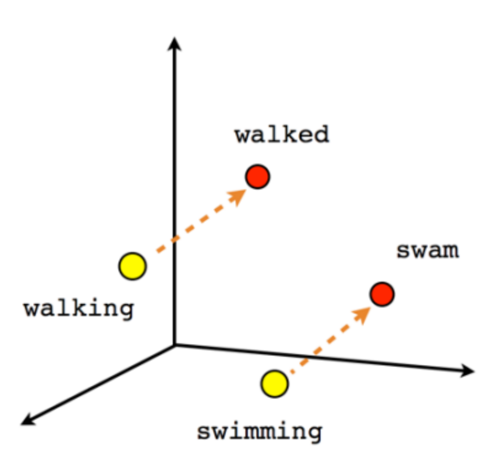
	Training Set	Dev Set	Test Set
Positive Reviews	12,500	6,250	6,250
Negative Reviews	12,500	6,250	6,250

PCA run over the dataset.



Features:

1. Bag-of-Words
   One-hot vector – size of the dictionary (400k words).
   Used for Naïve Bayes and SVM models.
2. Word Vectors [2]
   Pre-determined embedding in 50-dimensional space.
   Used for LSTM model.

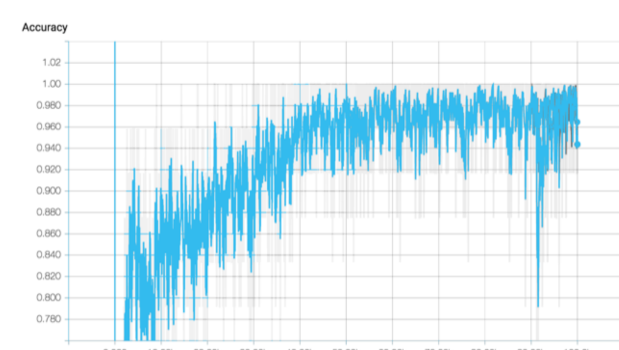


Future Work

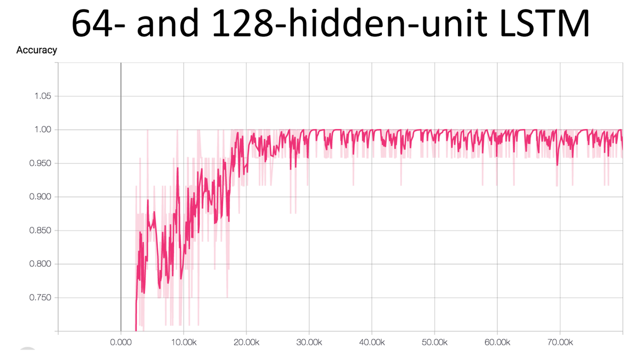
- Implement a deeper LSTM with mean-pooling layers
- Optimized memory allocation in TensorFlow code for JSMA method
- Adversarial training of LSTM network based on JSMA adversaries
- Use Stanford NLP Parser to automate grammar checking

Models

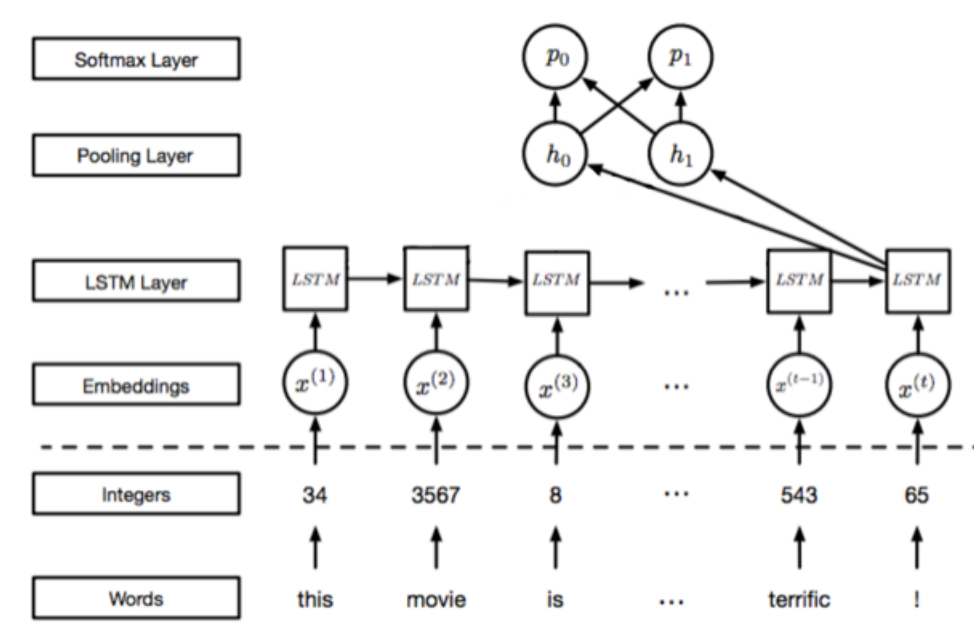
- Single-Layer RNN with LSTMs
- Linear SVM
- Naïve Bayes with Laplace Smoothing



Training accuracy vs. # iterations, 64- and 128-hidden-unit LSTM



The Word2Vec + LSTM architecture [3]

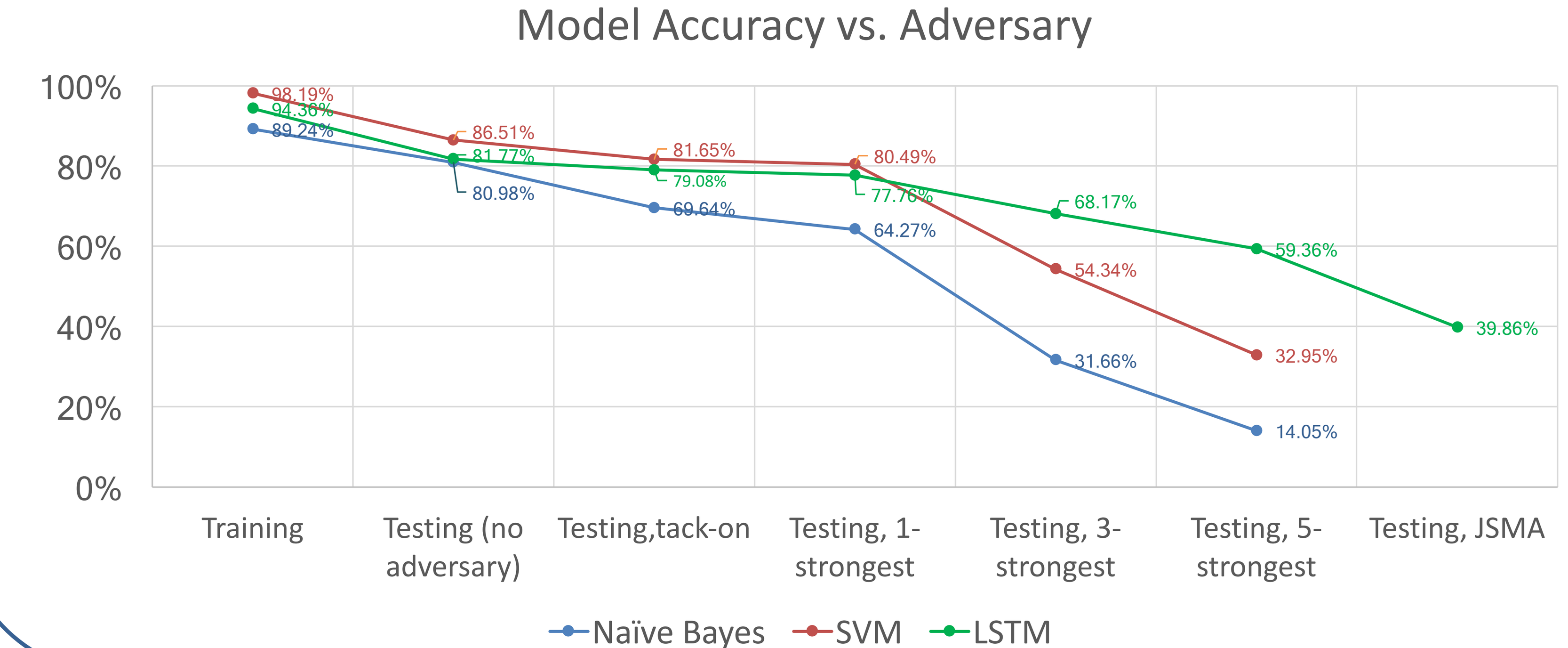


We performed a hyperparameter search and settled on an LSTM with a softmax output layer and 64 hidden units. For the linear SVM, we swept learning rate and tried different features and kernels. The Naïve Bayes model is multinomial and uses log-probabilities.

Analysis

- SVM and NB perform similarly to LSTM on the test set without adversary. This implies the data is well-segregated - independently seen in PCA plot.
- The LSTM is most robust to our black-box adversaries.
- Black-box adversaries were words strongly associated with sentiment.
- Model accuracies fell monotonically with increasing adversary strength.
- Jacobian-based methods do not always change the most positive/negative words. Seemingly-random word injection changes the prediction, leading us to question whether LSTMs are actually learning the sentiment; e.g.:
  This excellent movie made me cry! → this excellent tsunga telsim grrr cry

Model Accuracy vs. Adversary



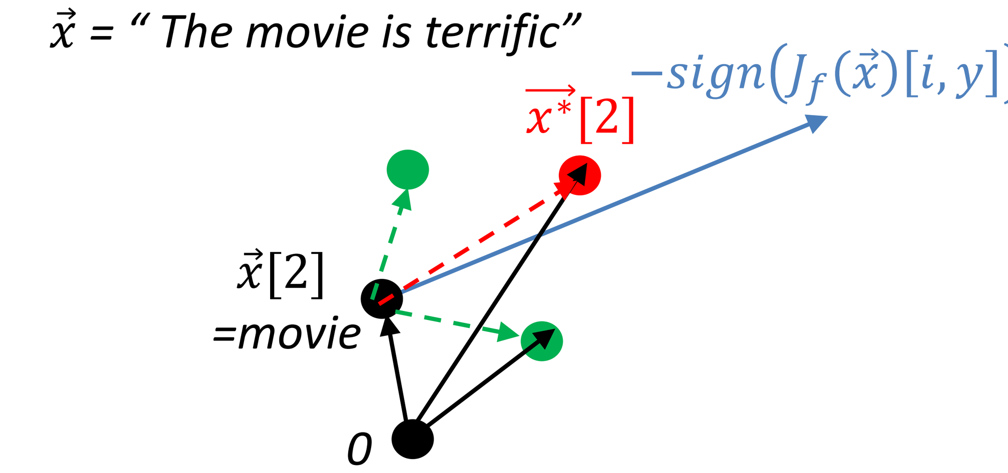
Intuitive Black-Box Adversaries

- Based on Naïve Bayes "strongest" words – words most polarizing toward positive or negative classification
- Adversarial Words:
  - Positive Sway: "edie," "antwone," "din," "gunga," "yokai"
  - Negative Sway: "boll," "410," "uwe," "tashan," "hobgoblins"
- Tack-On: replace first word with random adversarial word
- N Strongest-Word-Swap: replace review's N strongest word(s) with random adversarial word(s); experimented for N<=5

Jacobian Saliency Map Adversary [3]

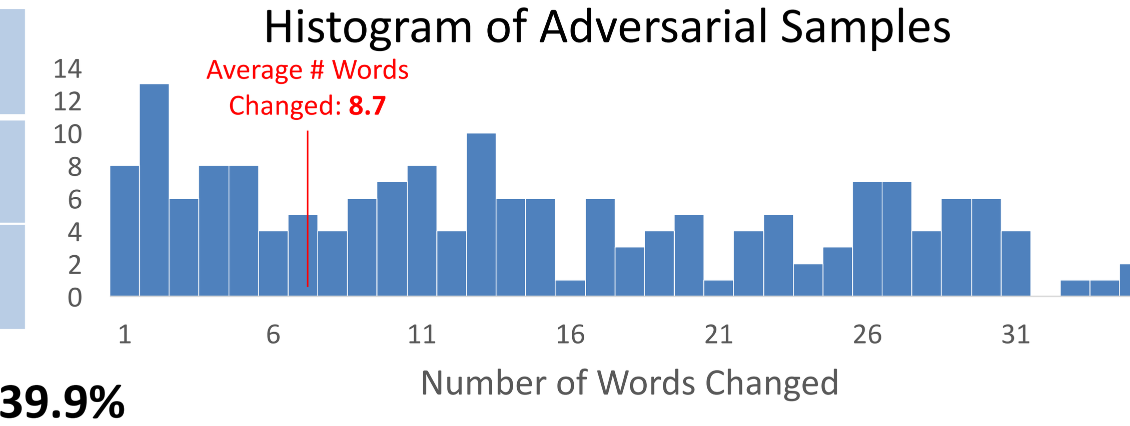
f: Prediction Model
X: Example Sentence
D: Dictionary

Input: $f, \vec{x}, D$
Algorithm:
1. $y := f(\vec{x})$
2. $\vec{x^*} := \vec{x}$
3. $J_f(\vec{x})[y] = \frac{\partial h_y}{\partial \vec{x}}$
4. while $f(\vec{x^*}) == y$:
5. select a word i in sequence $\vec{x^*}$
6. $\vec{w} := armin_{\vec{z} \in D} \left| sign(\vec{x^*} - \vec{z}) - sign(J_f(\vec{x})[i,y]) \right|$
7. $\vec{x^*}[i] := \vec{w}$
8. end
9. return $\vec{x^*}$



	y= Pos	y= Neg
True Positive	54	65
True Negative	110	62

Histogram of Adversarial Samples



Accuracy after JSMA = 39.9%

References

[1] A. Maas, R. Daly, P. Pham, D. Huang, A. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," In Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, '06, 2011, pp. 142-150.
[2] A. Deshpande, "Sentiment Analysis with LSTMs," Oct. 3, 2017. [Online]. Available: https://github.com/adeshpande3/LSTM-Sentiment-Analysis.
[3] N. Papernot, P. McDaniel, A. Swami, and R. Harang. "Crafting Adversarial Input Sequences for Recurrent Neural Networks." Apr. 28, 2016.