JDunn_Final_Report

December 8, 2019

Jimmy Dunn

1 Abstract

Sarcasm detection is an important part of sentiment analysis and relies on contextual and structural cues. In social media, particularly platforms like Twitter, the limitation of text length increases the importance of structural cues, like hashtags and emojis. This project explored the use of transfer learning emoji embeddings into a CNN machine model for detecting sarcasm in Twitter corpora. Due to an oversight in the design of the experiment, in conjunction with limited time and resources available to address it, I was unable to conclusively determine if pre-training a CNN with text and emoji embeddings could improve sarcasm detection in Twitter corpora stripped of non-word tokens.

2 Introduction

Sarcasm is the use of irony or satirical remarks, often tempered with humor, to expose absurdity in a topic or provide comic relief. Applied to text, sarcasm may be identified when the surface sentiment of a phrase differs from the actual sentiment intended by the speaker. While sarcasm has predominantly been used in verbal form and rarely seen in written form, the advent of social media has given rise to conversational written styles that become less formal and integrate more sentiment like sarcasm. Successful online communication is dependent on users understanding when sarcasm is being used, misinterpretations can lead to resentment between users or authors being affiliated with views they actually meant to mock.

Sarcasm detection is difficult task, not only for machine learning models but for human annotators as well. Modern research in detecting sarcasm in long form text focuses on contextual incongruity and lexical features. More cutting edge research is examining multimodal approaches to identifying sarcasm, combining image and speech recognition with textual context to identify sarcasm in TV shows and social media posts on platforms like Tumblr and Facebook. But these approaches still require rich contexts with large vocabulaies and long corpus lengths; tweets, by their very nature, have small vocabularies and limited corpus lengths, further complicated by a mix of words and non-word tokens.

Efforts to apply sarcasm detection techniques on Twitter has shifted the focus away from context to structured content, like hashtags. While multimodal approaches have been applied to social media, emojis haven't yet been explored as a feature. Emoji exploration is a great opportunity to apply a multimodal approach to sarcasm detection to Twitter. In this project, I apply a

multimodal sarcasm detection model on an annotated subset of Twitter data by comparing performance between a simple model trained on only word tokens and a model pre-trained on words and emojis.

3 Methods

3.1 Data

The primary data source I have chosen to test comes from Sentiment140, a project designed by a group of computer science graduate students at Stanford that uses distant supervision methods to classify the sentiment of tweets. The inspiration for this project comes from an assumption they programmed into their methodology: that the sentiment of a tweet containing a ":)" emoticon is always positive while a tweet containing a ":(" is always negative. The Sentiment140 training data set is a comma separated values file of tweet texts and polarity labels (0 - negative, 4 - positive). The tweet texts have been stripped of emojis and non-word tokens.

The secondary data source I have chosen for pre-training my model comes from DeepMoji, a project based in MIT that classifies a broad variety of sentiments in tweets with emojis, including sarcasm, emotion, and slang. I specifically use two of their benchmark data sets they created text and emoji embeddings from, the SCv1 and SCv2 data sets.

3.2 Hand Labeled Data

I decided to personally annotate 1000 randomly selected examples from the original corpus of 1.6 million tweets in Sentiment140. The positive label density was 2.5% or 25 examples. While I could've performed a string search for a specific pattern to increase the number of sarcastic examples, I didn't want to bias the model towards that search pattern. I decided to settle for a small positive label bucket to preserve randomness. Intuitively, real life sarcasm use on social media may actually by close to this percentage, or at the very least I can assume that sarcasm is not the dominant form of communication.

3.3 Tools Used

I used keras with a tensorflow backend as my primary technology stack for this project. I used the tflearn Vocabulary Processor as my embedding tool.

3.4 Embeddings

I preprocessed the sentiment data by treating each tweet as a sentence. I stripped special characters and converted words to lowercase. I split the tweets into tokens based on spaces between words. I fit these tokens into a tensorflow vocabulary processor object which I then used to pad all tweets to be the maximum sentence length. I then take these embedding vectors and split them in 80/20 training and validation subsets. I did not need to preprocess the SCv1 and SCv2-GEN dataset as the creators of DeepMoji made available the direct embeddings themselves as a serialized Pickle file. I loaded the embeddings directly for both DeepMoji datasets.

4 Model

4.1 Convolutional Neural Net

I feed these embedding vectors as inputs into a four layer Convolutional Neural Net I implemented in keras. The first layer is a convolutional kernal that transforms the inputs into a one dimensional output tensor using a relu activation function. The second layer is a global max pooling layer. The third layer is a densely connected layer using a relu activation function and the final layer is a densely conencted layer outputting to two dimensions using a sigmoid activation function. The model uses the adam optimizer and binary cross entropy to compute loss at each step.

To test my hypothesis that people who tweet use emojis in conjunction with specific words or phrases, I pre-trained the CNN on DeepMoji's sarcasm tweet dataset. I pre-trained two models, one on the SCv1 and one on the SCv2. I then treated the entire 300 tweet hand labeled corpus as a validation set and evaluated model performance.

Table 1 shows the comparison of parameters and training times for the baseline, SCv1, and SCv2 models. Training time is on the W266 issued Google Cloud compute instance with ~4GB RAM.

4.1.1 Table 1: Model Training Parameters

Model	Vocabulary Size	Training Examples	Total Parameters	Training Time
Baseline	3589	800	153,673	< 1s per epoch
SCv1	11084	1596	13,390,237	5 min per epoch
SCv2	11929	2608	2,578,074	26s per epoch

5 Results

Table 2 details the acccuray and loss generated by evaluating the classification of Tweets as sarcastic or not using the simple CNN and the pre-trained CNN.

5.0.1 Table 2: Model Accuray and Loss on Validation Data

Model	Validation Accuracy	Validation Loss
Baseline	0.9650	0.3217
SCv1	0.2050	3.4272
SCv2	0.0570	8.0562

The results show that pre-training the Convolutional Neural Network on the SCv1 data set dramatically worsens predictive performance on the validation set and the SCv2 pre-trained model is even worse. I suspect the reason that this experiment failed so spectacularly is because there is insufficient overlap between the vocabularies of the training data sets and the annotated validation data set.

Interestingly, the CNN model itself performs pretty well on the baseline, with a validation accuracy of 96.5% or 7 misclassified examples out of 200. The CNN model also performs well

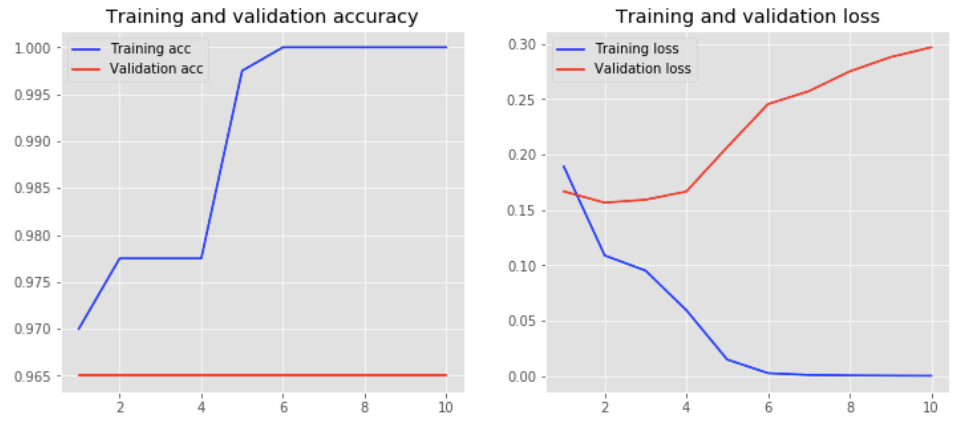

Figure 2: Baseline Training Performance

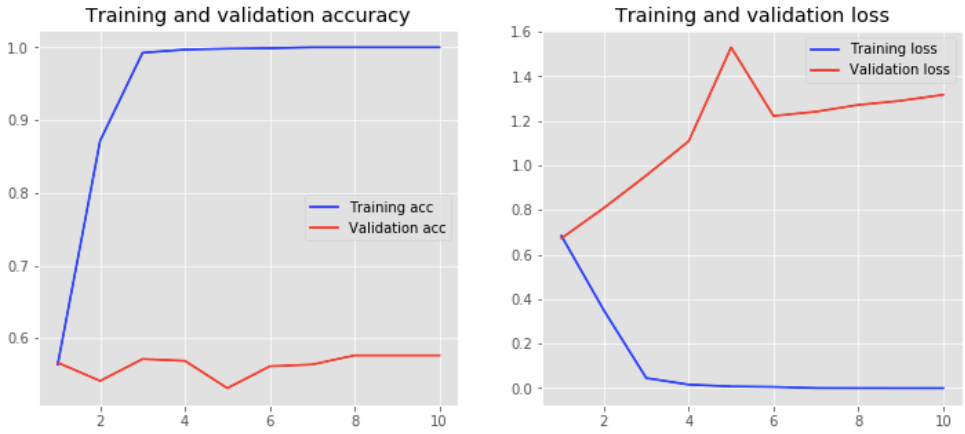

Figure 3: SCv1 Training Performance

internally on each data set. Figure 2-4 shows the comparison of internal training performance across all three models.

6 Conclusion

I acknowledge that I made several key assumptions in designing and implementing this project. I assumed that my annotations of what tweets are sarcastic or serious are accurate annotations. I did have some difficulty making decisions on some Tweets, because without emojis or other structured tags to indicate sarcasm I was left undecided. I also assumed that there would be enough overlap in vocabulary between the SCv1/SCv2 data and the Sentiment140 annotated subset for the predictions to be meaningful.

I cannot make any definitive conclusions about the utility of emoji tokens in improving sarcasm classifiction on Twitter corpora that has been stripped of non-word tokens. In order for me to confirm these negative results, I would need to have a much larger annotated corpus of sarcastic and non-sarcastic tweets, with the goal of increasing the vocabulary diversity of the validation set to improve overlap with the SCv1/SCv2 vocabularies.

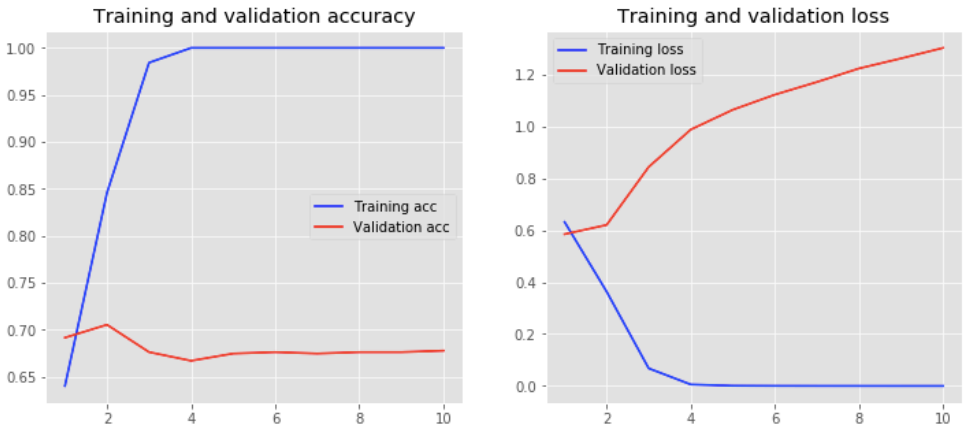

Figure 4: SCv2 Training Performance

7 Further Research

More Annotations: By increasing the number of annotated examples, with a better balance of positive and negative sarcasm labels, I might be able to improve performance of the transfer learning experiment. Unfortunately, that would require a much larger scale than just 2000 or even 3000 labeled tweets.

Better Embeddings: I used basic word indices as my embeddings. Given more time I would've liked to use more robust, pre-trained embeddings like BERT.

Attention Model: I began to explore the use of an attention model, composed of a bidirectional LSTM RNN, but did not have sufficient memory to run the model. If I was able to purchase a larger virtual machine, I would be interested in comparing performance between the CNN and the attention model. I've included the code for my attention model in the repo.

Word-Emoji N-Grams: As the embeddings are made directly available from DeepMoji, I am unable to search the Tweet text to identify word-emoji pairs. It would be good to be able to identify which emojis are used in conjunction with the terms identified as commonly sarcastic. I would be able to search through the Sentiment140 corpus much quicker for those terms without worrying about introducing my own personal bias when annotating.

Large Scale Evaluation: It also would be interesting to see if we could get human evaluators to evaluate semi supervised labeling of tweets, similar to what the original authors of Sentiment140 set out to achieve but with the sarcasm function enabled.

7.1 Thanks

Thank you to the developers of keras, tensorflow, and all of the open source contributors to the natural language processing domain that allowed me to pursue this project on my own.

Special thank you to Dan Cer, James Kunz, and the entire w266 teaching team for making my final term in the MIDS program a fun and challenging one.

8 References

Data Sources

Sentiment140: Twitter Sentiment Classification using Distant Supervision, Go et al. 2009

SCv1: A Corpus for Research on Deliberation and Debate, Walker et al. 2012

SCv2-GEN: Creating and Characterizing a Diverse Corpus of Sarcasm in Dialogue, Oraby et al. 2016

Code Sources

String Preprocessing: Convolutional Neural Networks for Sentence Classification, Yoon Kim 2014

Plotting History: Practical Text Classification With Python and Keras, Nikolai Janakiev

Methods for Detecting Sarcasm

Detecting Ironic Intent in Creative Comparisons, Veale and Hao 2010

Clues for Detecting Irony in User Generated Contents, Carvalho et al. 2009

Towards Multimodal Sarcasm Detection, Castro et al. 2019

Sarcasm in Social Media

Detecting Sarcasm in Multimodal Social Platforms, Schifanella et al. 2016

Contextualized Sarcasm Detection on Twitter, Bamman and Smith 2015

Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon, Davidov et al. 2010

Non-Sarcasm Sentiment Research

A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, Pang and Lee 2004

What Does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis, Barbieri et al. 2016