Are You Joking? Deep Learning to Identify Humor

Gene Ahn | Kiersten Henderson | Matt Stevenson

Abstract

Humor is viewed as a distinctly human trait. It is important to be able to detect humor and understand what causes it, so that humor can be recognized and emulated by machines. We built a classifier to detect humor and we then analyzed some of the underlying features of jokes that could cause humor. We used a Convolutional Neural Network to train a humor detection model on a labeled data source that contained funny one-liners and newspaper headlines that were not funny. Our model achieved 98.20% accuracy. However, because humor is often contextual, and because there are different kinds of humor, we applied our model to a second dataset to check transferability. This dataset was composed of different varieties of short jokes balanced against adages and news headlines that were not funny. Our model achieves 78.58% accuracy on this second dataset. Previously, both sentiment and uncommon word choices were hypothesized to cause humor. We therefore compared the average sentiment of jokes and non-jokes and we also replaced uncommon words in jokes with their most common synonym and looked for differences in prediction by our model. Although neither of our analyses of the causes of humor identified a feature associated with humor, we identify next steps for future exploration.

Background and Motivation

Humor is viewed as a distinctly human trait. In popular culture, robots and computers are often portrayed lacking humor even though they are highly proficient with other aspects of language. Understanding humor is important to achieving the complete contextual understanding of human communication. Current research seeks to advance the capabilities of humor prediction, with the goal of understanding the “causal elements” of humor (i.e., what makes something funny?). This knowledge can then enable success in the interpretation and generation of humor by machines.

Recent work (de Oliveira and Rodrigo, 2015), built binary classification models of funny and not-funny Yelp reviews. This work systematically tested the performance of shallow models such as Logistic Regression and Support Vector Machines compared with higher-order structures such as Recurrent Neural Networks and Convolutional Neural Networks (CNN). Additionally, they varied the sophistication of language models from simple Bag-of-Words representations to the vectorized representations of words and

phrases. This study found that CNN was best able to predict humor with 81% accuracy. A subsequent study (Chen and Soo, 2018) also favored CNN to implement in-text humor detection on a joke dataset with similar success.

While these studies achieved state-of-the-art accuracy in binary humor prediction of phrases, we are additionally interested in identifying features associated with humorous language. Thus, we created a CNN to predict humor and then examined features associated with humor. There are several hypotheses of the causal factors that dictate humor. Recent work suggests that uncommon word choice is correlated with humor. Additionally, previous work finds a correlation between sentiment (e.g., positive vs. negative) and humor. However, there are conflicting findings - some find that more positive sentiment correlates with humor, whereas others find that more negative sentiment is correlated with humor (reviewed in Shahaf, Horvitz & Mankoff, 2015).

We tested these hypotheses using two different approaches. First, to compare the overall sentiment of funny and not-funny phrases, we calculate average overall sentiment scores for each group of phrases and use statistical inference to determine whether they differ in sentiment. We created a sentiment classifier to enable this task. Second, to examine the effect of unusual word choice on humor, we identified the most unusual word in each joke and replaced it with its most common synonym. We also tried replacing every word with its most common synonym. We built a synonym-frequency thesaurus to enable this task.

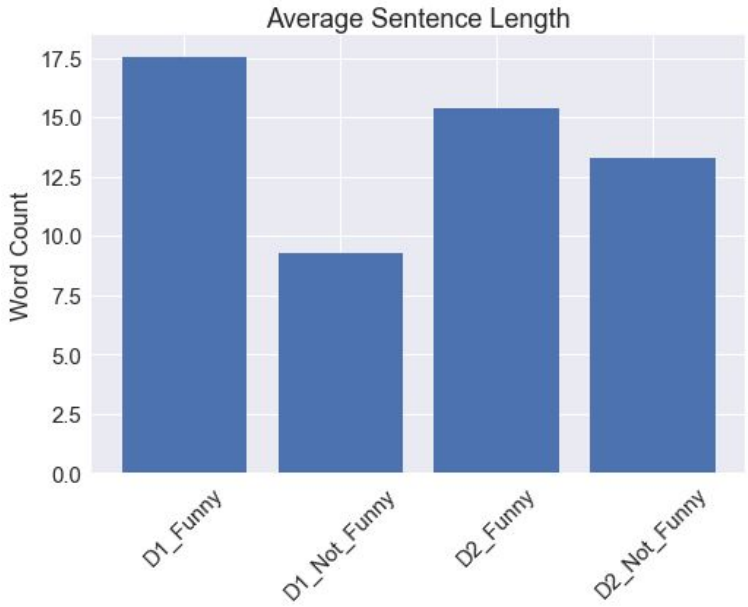
Methods

The code to implement the CNNs from previously published work was not available, though we were able to recreate a similar model to measure performance of humor detection in two contexts. Our implementation was sequenced through the following: (1) EDA of our first dataset, (2) data transformation, (3) training and scoring a baseline model, (4) training and tuning CNNs with our own word-embeddings, (5) training CNNs with pretrained GloVe embeddings, (6) generalizing across a different context of humor and non-humor, and (7) analysis of sentiment and word choice on humor.

Data

We assembled our datasets from several sources of labeled data available on Kaggle and CrowdTruth. All sources included a passage of text and binary labels for “funny” and “not funny”. We aggregated our sources into “dataset 1” and “dataset 2”, which we used for initial model training/test and further generalization, respectively (see Github).

Each dataset was had a roughly equal proportion of funny and non-funny examples. Dataset 1 consisted of 463K rows that were evenly balanced between (1) a collection of funny “short jokes” and (2) aggregated new article headlines sourced from a UC Irvine data repository. Dataset 2 consisted of 11K rows of text that spanned a broader range of formats and topics including “one liners”, longer jokes, Reuters headlines, proverbs, and Wikipedia sentences. In Dataset 1, the average length of funny sentences (in words) are twice as long as non-funny sentences. However, dataset 2 has more similar sentence lengths between funny and non-funny examples.



Additionally, we found little overlap between the top 20 most common words between Dataset 1 and 2. Only a single word (“says”) overlaps between the top 20 lists across both datasets. This indicates that the types of words that are used in each context (funny and non-funny) may be inherently different.

Below are examples for labeled “funny” and “not funny” data from Dataset 1:

- Funny: *telling my daughter garlic is good for you. good immune system and keeps pests away.ticks, mosquitos, vampires... men.*
- Not Funny: *fed chair janet yellen speaks of beefing up regulations*

Text Preprocessing for Logistic Regression and CNN

We maintained stop words and punctuation in the text because previous work found these contributed to successful humor prediction (de Oliveira and Rodrigo, 2015).

However, we did convert all text to lowercase prior to tokenizing. We found that the length of the text passages ranged from 1 to 1837 words with a strong right skew. More than 99% of training examples had a maximum document length of 34 words or less. Therefore, we set 34 as the max document length to allow for the appropriate padding of training examples in the CNN. Vocabulary processing and tokenization resulted in a vocabulary count of 133K unique words.

Baseline Model

Prior to training higher-order models, we began with a baseline prediction using a single-layer Logistic Regression applied to the training data of our first dataset. With L2 regularization, the baseline logistic regression model yielded an accuracy of **0.8160** and an F1 score of **0.8128** on the test data (see Github).

Baseline CNN

Previous work that used CNNs to predict humor was based on work that used CNN for sentence classification (Kim, 2014). We therefore built our CNN in a similar manner by following a tutorial that recreated part of this model in TensorFlow (Britz, 2015).

The first layer of our CNN embedded words into 300 dimensional vectors. The next layer performed convolutions over the embedded word vectors using multiple filter sizes (3, 4, 5 words). Next, we max-pool the results of the convolutional layer into a long feature vector, added dropout regularization, and classified the result using a softmax layer.

Final CNN

We tuned the CNN by experimenting with hyperparameters for dropout, learning rate, and number of filters and chose a final model that used the AdamOptimizer and had a learning rate of 0.0001, dropout = 0.5, and 100 filters (**See Table 1**). We did not experiment with L2 regularization, as in similar contexts to our work, its effects have been negligible (Zhang and Wallace, 2015). Our final model had an accuracy on the test set of **0.9820.**

Pre-trained Word Embeddings

We went on to include pretrained GloVe word embeddings (Pennington et al., 2014) in our modelling strategy, based on a tutorial (Li, 2017). We created our embedding lookup table including these embeddings and trained our model in both the situations where the pre-trained word embeddings were not trained, or where they were trained with the rest

of our parameters. The maximum accuracy on the dev set in either case was: 0.9266 for leaving them untrained versus 0.9469 for allowing them to train (see Github). Previous work in sentence classification achieved large gains in accuracy using pre-trained word embeddings, so we were surprised that we did not find similar results (Kim, 2014). However, the final model we created by training our own embeddings was very accurate (0.9820) on test data from our first data set, so we implemented it and continued our analysis (see Github).

Table 1: Evaluating Potential Models

Model	**Accuracy**
Logistic Regression, L2 Regularization	0.8160
Base CNN, embedding_dim = 300, filters = [3,4,5], fiter_number = 100, Adam_Optimizer, Learning_Rate = 0.01, dropout = 1.0	0.9645
Final CNN: embedding_dim = 300, filters = [3,4,5], fiter_number = 100, Adam_Optimizer, Learning_Rate = 0.0001, dropout = 0.5	**0.9820**
Final CNN + pre-trained GloVe embeddings (not trained in CNN)	0.9266
Final CNN + pre-trained GloVe embeddings (trained in CNN)	0.9469

Table 2: Error Analysis on Data Set 1

Category	**Joke Text**
TN	*older women are most vulnerable to cervical cancer*
TP	*i tried using self deprecating humor but i'm not any good at it.*
FN	*what was spiderman's major in college? web design*
FP	*so it turns out that dinosaurs were neither warm nor cold blooded*

Results

In general our final model was very good at making predictions on our first dataset (0.9820 accuracy). In addition the the model made fairly balanced mistakes in terms of classifications: 57% of misclassifications were false negatives and 43% were false positives. When we performed error analysis, a couple of caveats in our choice of data for dataset1 became apparent. Sometimes false positives are funny - for example: "*Julia Roberts Confronts the Dog that's Afraid of her Face*". This does point out a potential caveat in our dataset - we have assumed that news headlines are not funny without reading them, and labeled them as such, but some headlines will actually be funny or clever. And sometimes jokes that are labeled funny in our dataset are not actually funny: "*Bioluminescent Fry said to have a Bright Future*" is labeled funny, but instead sounds more like a news article headline. This points out that humor is subjective and the dataset has been labeled by many different people with different opinions who created the jokes. If the is joke merely clever instead of LOL funny, some people might still label it as funny.

How Well Does our Model Generalize?: Applying our model to a Different Dataset

In order to get a sense for how well our model generalizes to different kinds of humor and different kinds of non-humorous text, we made predictions to a second labeled dataset that we obtained from Kaggle and cleaned (described above). As mentioned above, this dataset contained a larger variety of jokes and the non-humorous text was composed of adages as well as news article headlines.
Although our model was far less accurate on this second dataset (0.7858), this is very good performance considering how difficult humor is to identify. As a benchmark comparison, the leading CNN designed in previous studies had ~82% accuracy on Yelp review test data (de Oliveira and Rodrigo, 2015).

Table 3: Evaluating the Final Model on Dataset 2 and on Altered Versions of Dataset2

Dataset scored with Final CNN	Accuracy
Second dataset	0.7858
Second dataset, least frequent word replaced with most frequent synonym	0.7858
Second dataset, all possible words replaced with most frequent synonym	0.7514

However, whereas the model made fairly balanced errors on dataset 1 (57% errors false negative, 43% false positive), it makes a whopping 94% of mistakes as false positives on the second dataset. This suggests that the adages used as non-funny phrases may be too similar to humor for our model to distinguish between them. For instance the model erroneously classifies "*He is not fit to command others that cannot command himself*", "*Truth is stranger than fiction*" and "*Well begun is half done*" as funny. In addition, there appear to be a higher percentage of non-funny phrases that contain scientific language and fewer non-funny phrases are about popular culture (containing the names of celebrities) than in dataset 1. These findings underscore how important it will be in future work to have a very diverse set of labeled negative (non-funny) training examples.

Evaluating How Sentiment Correlates with Humor

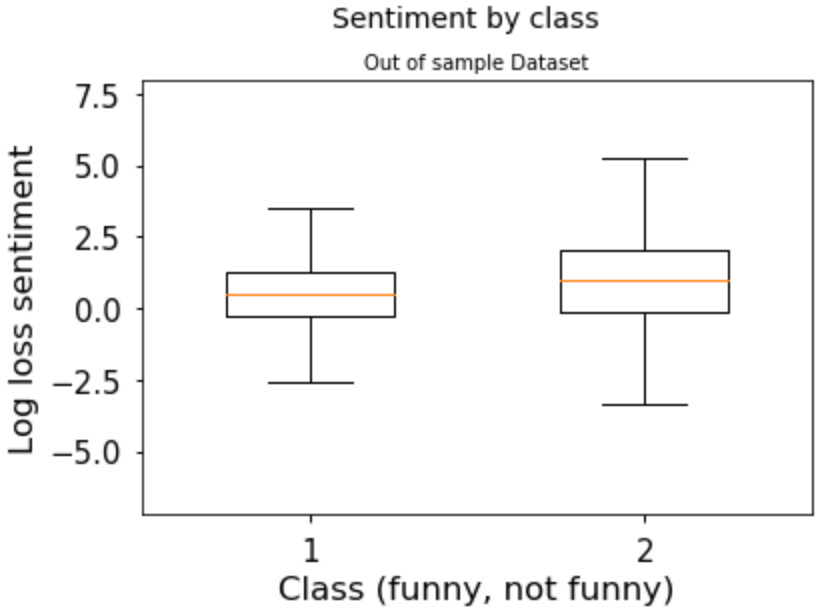
In order to investigate whether sentiment was a good indicator of funniness in our transfer dataset, we built a sentiment classifier (see Github). To do so, we append pretrained Glove embeddings to a large labeled set of positive and negative words from Hu and Liu (2004). Using a basic stochastic gradient descent classifier in SciKitLearn, we achieve 95% accuracy predicting log loss on single words in a held out set. The relative simplicity of this model is useful for making sense of the log loss sentiment scores (Speer, 2017). Using this sentiment classifier, we predict the average log loss sentiment for every text example (funny and not-funny) in our second dataset.

While the classifier does quite a good job scoring simple sentences, it has a harder time with longer or more convoluted sentence structures, generally dumbing down those scores to have scores closer to 0 (neutral sentiment). However, since we kept all stopwords and punctuation in the original CNN model as indicated in prior research, we retained these during our sentiment analysis. For example, "*Good health is above wealth*" scored an appropriately positive 5.9, whereas the quite negative "justice department considers legal changes to combat domestic anti-government extremists" scores -0.6, despite referencing a number of very negative concepts.

We did not normalize the sentiment scores by length of the sentence/joke, as words with large sentiment weights did not greatly affect the average sentiment within the range of 5-20 words, encompassing the majority of documents.

One theory of humor is that humor is often related to usage of words that are loaded with high sentiment value (Liu et al., 2018). In our case, the sentiment predictions do not

necessarily match the tone of the joke, which could indicate contra-positive usage of sentiment in jokes. For example, while a simple joke such as “*Bad becomes good when worse happens*” scores firmly in the negative with an average log loss of -3.25, "*I'd kill for a nobel peace prize*" gets a log loss of 3.78, putting it in the top 50 most positive jokes despite a very negative sentiment as judged by a human. Manually scoring 100 randomly sampled documents, the average sentiment classification matches up with the actual human-read sentiment (negative, neutral, positive) about 35% of the time. A method of sentiment analysis that has longer term dependencies rather than averaging based on word content is warranted for the detection of sentiment, however this is outside the scope of this paper.



As seen in the chart above, while the log loss sentiment predictions were in the same range for both classes, the variance of the average sentiments differs pretty heavily. Given that the sentiment frequency curves are close to normal for both classes, using a Welch’s T-test to test the similarity of the sentiment score distributions between the two classes funny and non-funny, we find a significantly different distribution, which is not surprising given that n= 5434 for each balanced class. In reality, however, the averaging technique employed to calculate sentiment drives the average log-loss score quite close together, with $\mu = .47$ (s = 1.21) for the funny class, and $\mu = .91$ (s = 1.69) for the not funny class. Both distributions hover around 0, and given the much wider variance (40% greater) of sentiment in the not funny class, this trend refutes our hypothesis that jokes would contain higher variation in sentiment.

Evaluating the Effect of Uncommon Words on Detection of Humor by our Classifier

To validate whether uncommon words (that are potentially present in funny examples) have an impact on our predictions, we transformed dataset 2 to replace uncommon words found in funny examples with their most common synonyms. We used the Synset module from Wordnet, an NLTK feature that contains sets of semantic synonyms for English words (see Github).

We took the unique vocabulary list from our funny examples and built a dictionary with the unique word as the key and a set of possible synonyms as the values. To determine the frequency or recurrence of each unique word and their synonyms, we found how frequently each word appeared in the combined Brown and Reuters corpuses.

Then, we replaced words from each funny example with its most common synonym. If the existing word was more common than any of its synonyms, we kept it. However, this often led to transformations that ignored part of speech, context, and lost semantic meaning for jokes that rely on multiple interpretations of the same word. For example, the original joke, “a cement mixer collided with a prison van. people are warned to be on the lookout for 15 hardened criminals” was transformed into “a cement social jar with a prison van. people be discourage to be on the outlook for 15 set criminals.” To mitigate this, we also created another dataset that replaced only the most uncommon word from each example with its most common synonym.

We then applied the model to predict humor on the modified dataset and found that accuracy decreased to 0.7514 (-0.034 vs. non-modified dataset 2) when all words were replaced with their synonyms and that accuracy remained unchanged at 0.7858 when only the most uncommon words were replaced (see Table 3). While these results of the first test directionally supports the notion that there may be a correlation between uncommon words and humor, we believe that we need more sophisticated transformations to retain the original semantic meaning of the text.

Next Steps

Throughout the literature, the main limiting factor for accurately judging humor is often noted to be the paucity of good datasets available. Indeed, while this paper relies upon lists of jokes paired with non-jokes, there is no guarantee that the jokes are funny or that the headlines, etc, are not. In truth, the best we can do with this data is to predict what is a joke and what is not. Acosta (2016) takes the approach to scrape the transcripts of

Ted talks and tag audience laughter, adding a human-confirmed response variable as the binary label. However, the approach suffers from long term dependency as what is said immediately before the laughter is not necessarily the punch line. Longer term dependencies are much harder to model, even when using techniques such as the CNN applied herein. Other, non binary response variables such as measuring intensity of laughing, or having a number of reviewers rate jokes as funny or not and taking the average score are also possible, and would add great value to the NLP pursuits in this domain of humor.

In the context of our paper, despite the low difference in sentiment difference between the jokes and non-jokes, more sophisticated sentiment analysis within the joke itself might be warranted. One avenue would be take into account the semantic difference (cosine distance between word embeddings) among words in a given sentence that differed heavily in word sentiment score (Liu et al., 2018). If found, experimenting with the effects of altering sentiment by switching out highly sentient words for their less-sentient or opposite-sentiment synonyms would be of interest to interrogate the theory that jokes rely on contrasting sentiment to express humor.

In order to take our analysis of synonyms further while retaining meaning in the document, the next logical step is to match the part of speech of the word to be replaced with a synonym of the same part of speech. We have not seen this replicated in the literature to date, and would be an interesting avenue of inquiry to develop.

References

Abhinav Moudgil (2018) **Short jokes: Collection of over 200,000 short jokes for humour research.** Kaggle. https://www.kaggle.com/abhinavmoudgil95/short-jokes

CrowdTruth (2018) **Short Text Corpus For Humor Detection**. https://github.com/CrowdTruth/Short-Text-Corpus-For-Humor-Detection

Luke de Oliveira and Alfredo Lainez Rodrigo. (2015) **Humor detection in yelp reviews.** http://web.stanford.edu/class/cs224d/reports/OliveiraLuke.pdf

Peng-Yu Chen and Von-Wun Soo. **Humor recognition using deep learning.** (2018) Proceedings of NAACL-HLT 2018, pg 113–117. http://aclweb.org/anthology/N18-2018

Yoon Kim. (2014) **Convolutional Neural Networks for Sentence Classification.** EMNLP. https://arxiv.org/abs/1408.5882

Lei Chen and Chong MIn Lee. (2017) **Predicting Audience's Laughter Using Convolutional Neural Networks.** ArXiv e-prints:1702.02584.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. (2015) **Inside Jokes: Identifying Humorous Cartoon Captions.** KDD '15 Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pg 1065-1074

Chris Westbury, Cyrus Shaoul, Gail Moroschan, and Michael Ramscar. (2016) **Telling the world's least funny jokes: On the quantification of humor as entropy**. Journal of Memory and Language, Volume 86, pg 141-156.

Minqing Hu and Bing Liu. (2004) **Mining and summarizing customer reviews.** In *Knowledge Discovery and Data Mining*.
https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

Robyn Speer. (2017) **How to make a Racist AI without really trying.** Blog post on ConceptNet.
http://blog.conceptnet.io/posts/2017/how-to-make-a-racist-ai-without-really-trying/

Jeffrey Pennington, Richard Socher, and Christopher D. Manning (2014) **Glove: Global vectors for word representation.** Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. https://www.aclweb.org/anthology/D14-1162

Ye Zhang and Byron Wallace. (2015) **A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification**. IJCNLP.
https://arxiv.org/abs/1510.03820

Andrew Acosta. (2016) **Laff-O-Tron: Laugh Prediction in TED Talks.** Master's thesis, California Polytechnic State University, San Luis Obispo, CA.
https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=2849&context=theses

Lizhen Liu, Donghai Zheng, and Wei Song. (2018) **Modeling Sentiment Association in Discourse for Humor Recognition.** Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 586–591.
http://www.aclweb.org/anthology/P18-2093

Denny Britz. (2015) **Implementing a CNN for Text Classification in TensorFlow.** Blog Post from Wild ML: Artificial Intelligence, Deep Learning, and NLP.
http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

Irene Li. (2017) **TensorFlow 07: Word Embeddings (2) - Loading Pre-Trained Vectors.** Blog Post from Cafe, Bonne Nuit. https://ireneli.eu/2017/01/17/tensorflow-07-word-embeddings-2-loading-pre-trained-vectors/