Deep Humor Recognition

N. Lee
Matthew Liechty

Introduction

Detecting humor is important in order to understand natural language. Jokes often have very different meaning if read seriously. Sometimes the meaning is completely opposite! Therefore, determining whether there is humor in text is important to consider when automatically extracting semantics from text. However, identifying humor in natural language can be difficult. Humor is often subjective and many times relies on distant or unstated information. Jokes often involve new or subtle word semantics. Therefore, these substantial obstacles test the common informal hypothesis that any objective for which there exists a lot of labeled data can be learned using machine learning. Some people would say that people are born with a sense of humor (or not) and therefore it should not be able to be learned by a machine. We endeavor to show that humor is at least somewhat learnable by machines. A current theory says that humor is characterized by semantic incongruity.[9] Therefore, the deep learning algorithms that use a word embedding should perform better than those that do not as well as the 'shallow' machine learning algorithms. On top of this, we are interested in determining whether the varieties of RNN or CNN's perform better at this task.

Our project was inspired by a paper written by Stanford students titled "Humor Detection in Yelp reviews".[1] It tried to predict whether a Yelp review was humorous using a variety of machine learning algorithms including Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs).

Data

Just like the Stanford paper we used the Yelp Challenge Dataset with its millions of Yelp reviews which users have upvoted as "funny", "helpful", or "cool".[2] Just like this paper, we chose to count those reviews with 3 or more "funny" upvotes as our positive label with the rest being negative. We chose 3 reviews so we could be confident that users found this review funny. Unfortunately, this assumption adds noise to our dataset since reviews that are actually funny

could have been labeled with 1 or 2 upvotes or none at all if no one saw it. However, for this experiment, we needed to choose a threshold. With this assumption implemented, there was approximately a 90:10 negative to positive label ratio in the dataset. However, after discussing it with the class, we decided that the classifiers would probably run better with a less extreme ratio. The paper balanced the number of positive and negative labeled samples in their dataset making it 50:50, but we decided to use all the positive labels and sample the rest of the dataset to create a 70:30 negative to positive ratio of labels. This split (as opposed to 50:50) gave us more data to work with and is also closer to the natural distribution of humor in Yelp reviews. After getting the data in our desired format, we decided to use about ½ million examples for training. We tokenized the text using NLTK,[3] lower-cased all the words, and replaced all words with only a single occurrence in the entire corpus with an "<unk>" token. We decided to truncate text over 150 words and pad text under 150 words. For word embeddings we decided to download and use the 300 dimensional GloVe embeddings based on the 6 billion word Wikipedia corpus.[4]

Methods

In order to compare with our proposed deep methods, we used several 'shallow' algorithms implemented in Sci-kit learn.[5] These used the bag of words vectorization method. We used Keras to implement several deep learning algorithms: CNN, BiGRU, BiLSTM, and a feed forward neural network. We used a few different sources to help us implement the CNNs[10], GloVe embeddings,[11] and F1 Score.[12] We decided to use bidirectional RNNs instead of the unidirectional versions that the Stanford paper used because we guessed they would perform better with more context. All the deep learning algorithms used static GloVe word embeddings (except where noted), ReLU as the activation on the hidden layers, sigmoid as the activation on the final layer, model vector length of 100, binary cross entropy loss, a learning rate which decayed by the learning rate divided by the epoch number, the adam optimizer, and evaluation using F1 score using a validation dataset consisting of 10% of the training data during training. In addition, the CNN used kgrams of length 3 and a filter count of 100. The table below shows the parameters specific to each algorithm.

Parameters for the deep learning algorithms

	Layers	Dropout	Dropout Layers	Learning Rate	Epoch
FFNN	4 (Flatten, Dense 500, Dense 100, Dense 1)	0.3	2	0.00001	20
BiLSTM	5 (BiLSTM, BiLSTM, Flatten, Dense 100, Dense 1)	0.2	3	0.00001	1
BiGRU	4 (BiGRU, BiGRU, Dense 100, Dense 1)	0.2	3	0.001	5
CNN	4 (Conv1D, Max Pooling, Dense 50, Dense 1)	0.2	2	0.001	15

Results

The resulting F1 scores for each of the algorithms is shown below. The calculated F1 scores for guessing the majority prediction and for a random prediction are added for comparison. We did not find any other papers that worked on a 30:70 positive to negative split of this particular dataset to compare our performance to so we also included balanced accuracy [7] on some runs. All of these algorithms performed better than random prediction with the deep learning algorithms generally performing better than the shallow algorithms. The BiLSTM and CNN performed about the same.

We tried to improve performance by allowing the word embeddings to be updated during training. However, with all the other settings the same, the performance was about the same. We also tried using static random word embeddings as a baseline, and their performance was only a few points worse. The feed forward neural network initially was learning but then gave a random output by the third epoch and never recovered. We also show an example confusion matrix and example text.

Shallow Algorithms

Model	**F1 Score**
Majority Prediction (100% negatives)	0.0
Random Prediction (50% pos 50% neg)	0.375
Random Forest depth=10, trees=200	0.4337
SVM C=1	0.4601
Decision Trees depth=10	0.4615
Logistic Regression C=10	0.4895
Multinomial Naive Bayes alpha=10	**0.5544**

Deep Algorithms with static word embeddings

Model	F1 Run 1	F1 Run 2	Bal Acc Run 2
Majority Prediction (100% negatives)	0	0	0.5
Random Prediction (50% pos 50% neg)	0.375	0.375	0.5
Feed Forward 500>100>1	0.4425	0.4639	0.6316
2-Layer BiGRU with 100 Dense	0.5405	0.5181	0.6665
2-Layer BiLSTM with 100 Dense	0.5703	**0.5820**	0.7019
CNN kernel=3 with 100 Dense	**0.5725**	0.5620	0.6894

Deep Algorithms with dynamic word embeddings

Model	F1	Bal Acc
Feed Forward 500>100>1	0	0.5
2-Layer BiGRU with 100 Dense	0.5596	0.6864
2-Layer BiLSTM with 100 Dense	**0.5649**	0.6926
CNN kernel=3 with 100 Dense	0.5574	0.6798

Deep Algorithms with random static word embeddings

Model	F1	Bal Acc
Feed Forward 500>100>1	0.5015	0.6489
2-Layer BiGRU with 100 Dense	0.5487	0.6811
2-Layer BiLSTM with 100 Dense	0.5158	0.6670
CNN kernel=3 with 100 Dense	0.5416	0.6709

BiGRU Confusion Matrix (from a different run; included to approximate original confusion matrix)

True Negative	38425
False Positive	10439
False Negative	4042
True Positive	7902

Examples from the BiGRU (from a different run; included to approximate original model)

	Review snippet
True positive	“The first thing I want you to know about this place is that my eighty-six year old grandmother-in-law gave it her stamp of approval and said that we are coming back. I don't know if it was the two margaritas she had that led to this remark, but the rest of the table agreed. Happy hour works well if you order too much food and don't want to feel guilty about it. Yeah, I'm a guy and am talking about paying for food that I was eating after I was full because I ordered it. Isn't that an obligation if you want to remain a man?...”
False positive	“We popped into the Valley Bar, while exploring phx celebrating our friends 1 year anniversary from her divorce...This place was probably the coolest spot we encountered while in downtown phoenix - I mean it's in a basement ... literally underground! It felt like the kinda spot you may find in a big city like NY or LA. It was a very chill vibe…”
True negative	“I don't recommend RENCO Roofing/ I was assisting my brother in getting bids to replace an old foam roof on his newly purchased home. I met with a salesman (Jari Marjanen) in September; He seemed knowledge and advised that he had 30 years of roofing experience. He made suggestions of how things should be done, including replacing two skylights, insulating the rooftop AC…”
False negative	“Pure Foot Spa, is my favorite place to get foot massage, they've made my friend yelp, cry, and holler out in pain. But who cares cause the massage are just good or even better!...”

Conclusion/Discussion

Our original hypothesis that humor is at least loosely learnable by a classifier was supported by our results. While we can say these algorithms were able to predict many examples of humor, it is difficult to say what aspects of humor they were identifying. For example, reviewers may use a different type of humor while writing reviews than when they post funny tweets. More work would need to be performed to determine what type of humor is present in Yelp reviews and how well models trained on this dataset generalize to other datasets.

Originally, we based our models off of the F1 metric. However, because the F1 score was not generalizable across datasets [6] and was hard to interpret intuitively, we reran the models to include the "balanced accuracy" metric.[7] The reason why the F1 score is hard to generalize is because its values range greatly depending on the balance of the dataset; for example, random predictions can produce an F1 score of 0 to ⅔ depending on the balance of the labels. Balanced accuracy can be interpreted in the same way as accuracy, making our results more comparable to those of other paper's. With our best results capping at about 70% balanced accuracy, we can confidently say there is room for improvement. The results from the Stanford paper reached up to 80% accuracy,[6] but their results do not have as much meaning because the ratio of their examples was highly synthetic. In reporting these results, we must acknowledge that the Yelp dataset is not ideal for learning humor. Our underlying assumption that a review is funny if and only if it receives three upvotes or more is hugely simplifying and is prone to noise. While it is certainly possible more information could be learned from this dataset using other methods, in a better dataset, each example would be carefully reviewed by multiple people before determining the label. On Yelp, some reviews may be seen by fewer people and therefore may receive fewer upvotes for its tags. Further, a review may be seen by many users, but not upvoted as funny if another aspect of the review was more prominent. In other words, there is only a limited amount of information that classifiers can learn.

Our results show that, in general, shallow algorithms did not perform well, especially linear classifiers and decision trees. However, the Multinomial Naive Bayes performed unexpectedly well for some reason. It rivaled the deep learning algorithms with their ensembles of neural networks, showing that there was enough information in bag of words alone to classify pretty

competitively. However, in general, deep methods performed much better besides the feed forward network.

We also tried updating the pre-trained GloVe embeddings during training. Dynamic word embeddings tailor the word embeddings to the vocabulary of the dataset. Since humor often involves new words or subtle word play, dynamic word embeddings should have improved performance. But they did not; this could mean that a concept for humor is being learned that uses common language. It is also likely that the information that would have been stored in the word embeddings was destroyed with the removal of the words that only occurred once in the corpus. It seemed that feed forward neural networks are not designed for dynamically training word embeddings. We did however, see consistently higher performance than most shallow classifiers when using the BiLSTM, BiGRU, and CNN, regardless of the word embedding configuration used, showing the value in using these networks in binary classification.

One surprising observation we made was that, many times the deep classifiers performed better with only a few epochs rather than a large number of epochs. This is something we did not expect and cannot completely explain. However, it seems that as the classifier further fits the training dataset, it performs slightly worse on the validation set. We know this is akin to overfitting, but it still occurred with up to 50% dropout between each layer. So this is something that needs further exploration. It is also possible that the differences in results between classifiers and types of embeddings could be related to the number of epochs they were trained on as the validation F1 score went up and down throughout training. The variation was such that the validation set could get a 0.58 score on epoch 3, a 0.52 on epoch 6, and back up to a 0.57 on epoch 9, so the final score depended a lot on what epoch it ended on. So in general, much variation among results can be explained by this. Therefore, we cannot make any confident assertions about how different configurations of word embeddings worked differently. It is worth noting that run 1 for deep algorithms with static word embeddings attempted to negate this variation in results by re-running the models until the model achieved a higher F1 score based on the observed fluctuations in the validation score among epochs. Realising this may have biased the model to the test set, we performed another organic run without this bias (run 2) to show a more natural result, which turned out to be similar nevertheless.

One of the interesting questions this type of study touches is whether RNNs and CNNs perform better for particular NLP tasks. Historically, RNNs and CNNs have performed better on different tasks.[8] For this particular experiment, which would be classified as “sentiment analysis”, we found that the two types of classifiers generally performed about the same.

These results show promise, but there is much more work that could be done to improve on them. For example, Yelp reviews tend to be rather verbose; further work should explore whether longer or shorter reviews can be learned more easily or accurately. Further, since we are fairly new to training these sorts of classifiers, there are likely other hyperparameter combinations or layer architectures that would perform better. Furthermore, newer types of word embeddings such as ELMo and BERT may show more noticeable performance increases. Another compelling experiment to try would be to run this dataset using an attention model. It makes intuitive sense that the subtleties of humor rely on particular aspects and combinations of words. An attention model would probably be quite good at capturing these patterns. We hope that further work will be done in this area so that eventually systems like this will be able to reliably extract the true sentiments contained in text.

References

1. Humor Detection in Yelp reviews. https://cs224d.stanford.edu/reports/OliveiraLuke.pdf
2. Yelp Dataset Challenge. https://www.yelp.com/dataset/challenge
3. Natural Language Toolkit. https://www.nltk.org/
4. Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/
5. Sci-kit Learn. https://scikit-learn.org/stable/
6. Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF). Journal of Machine Learning Technologies. 2 (1): 37–63. https://bioinfopublication.org/files/articles/2_1_1_JMLT.pdf
7. Brodersen, K.H.; Ong, C.S.; Stephan, K.E.; Buhmann, J.M. (2010). The balanced accuracy and its posterior distribution. Proceedings of the 20th International Conference on Pattern Recognition, 3121-24. http://ong-home.my/papers/brodersen10post-balacc.pdf
8. Comparative Study of CNN and RNN for Natural Language Processing. https://arxiv.org/pdf/1702.01923.pdf
9. Identifying Humor in Reviews using Background Text Sources. www.aclweb.org/anthology/D17-1051
10. Another Twitter sentiment analysis with Python — Part 11 (CNN + Word2Vec) by Ricky Kim. https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-11-cnn-word2vec-41f5e28eda74
11. Importing GloVe Embeddings into Tensorflowby Joseph Guhlin https://josephguhlin.com/importing-glove-embeddings-into-tensorflow/
12. How to calculate F1 Macro in Keras? Top answer. https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras