Humor is an abstract, high-level use of language that is largely subjective. Still, there are structures and patterns of language that are widely recognized as funny.

We attempt to build a model capable of recognizing humor within text. Specifically, we’re interested in humor as it arises unforced in everyday situations – that is, not scripted like a sitcom.

While a light-hearted endeavor, this work has potential applications in making personal assistants more responsive and life-like.

### Baseline
- The baseline algorithm consisted of a 2-layer neural network that took in a sum of 256-dimensional word vectors as the input. Used extensions like tf-idf and removing punctuation.

### CNN
- Earlier method loses order information, but RNNs can be slow. Made a CNN that takes a sentence as a stack of word vectors and performs convolutions with different windows before max-pooling the results and feeding into fully connected layer.

### Attention RNN
- Wanted to take advantage of sequential strengths of RNN but also modify the network to perform attention all hidden outputs and pass through softmax to give weightings to the hidden outputs to add together and pass to fully connected layer.

### LSTM-CNN
- Pass embeddings to LSTM network to obtain hidden outputs and run previous CNN on top of the stacked matrix of hidden outputs.

### Analysis
- TF-IDF improved the baseline, so the important words did indicate humor. Punctuation improved the baseline as well, since it helped show overreaction and transitions (! and ,).
- Severe overfitting because of high test accuracy/low test accuracy. Caused because humor in training set was direct story-based, and was sarcasm in test.
- Example: “I looooove eating sushi in dingy apts.”
- LSTM-CNN performed best with sarcasm (huge number of parameters to fit to model well enough).

### Results
- Evaluated with accuracy, precision, recall, F1
- The baseline returned about a 0.72/0.56 train/test accuracy. Variations such as using TF-IDF, and treating punctuation as separate embeddings yields 0.78/0.62 train/test accuracy.
- Approximately 0.8/0.6 train/test accuracy except LSTM-CNN had 0.9/0.7 train/test accuracy.
- F1 increased in order of Baseline, CNN, RNN, LSTM-CNN.

### References
- Sosa, “Twitter sentiment analysis using combined lstm-cnn models,” June 2017