Predicting Funny Yelp Reviews

Christina Pan
capan@stanford.edu

John Martin Poothokaran
johnmp@stanford.edu

Abstract

Deep Learning in the realm of Natural Language Processing (NLP) has made leaps and bounds in the field of sentiment analysis, yet there has been little work done in classifying humor. This paper illustrates a funny/not-funny binary classification experiment on the Yelp Review dataset based on a reviews' funny votes. In this paper, we plan on using a more complex neural architecture inspired by the bi-attentive classification network (BCN) by McCan et. al. Using a modified BCN architecture resulted in a F1 of 26.4.

1 Introduction

Humor is a wonderful facet of human communication that has not been deeply explored in sentiment analysis. For this project, we would like to build a model to perform sentiment analysis and predict the funny reviews as labeled in the Yelp dataset. We will use the dataset provided by Yelp [1], which features customer reviews with the number of votes for characteristics like cool and funny. This task of prediction involves several challenges, such as the skew in the amount of funny reviews, the size of the data, and the presence of misspelled words and slang.

In addition, we plan on approaching this problem using deep learning and NLP-specific deep-learning techniques. We aim to implement more complex architectures to test how well it can classify funny reviews in the dataset. An self matching attention model is implemented based on a bi-attentive classification network (BCN) introduced in McCan et. al. [2].

2 Background and Related Work

The idea of identifying humor in natural language corpora has been explored previously by Bertero and Fung [3], and Oliveira and Rodrigo [4]. Bertero and Fung set out to model the setup - punchline structure of jokes from the corpus of script and audio obtained from the comedy series - Big Bang Theory. An LSTM is used to model this structure and predict the punchlines from the audio data containing canned laughter, and the scripts from the series which provide the lines and the character that delivers them. In their paper, Oliveira and Rodrigo perform a binary classification of yelp reviews as funny/not funny and use a balanced set of funny/not funny reviews to train their models.

However, all of these papers' models, when featuring deep learning, only included straightforward RNNs or LSTMS. For our paper, we plan on focusing on a more complex architecture inspired by the bi-attentive classification network (BCN).

The BCN described by McCann et al [2] can handle one and two sentence classification tasks. The model described in the paper consists of a ReLU network through which the word vectors for sentence classification tasks are propagated before going through the encoder, which is a layer with bidirectional LSTM units. The output of the encoder for each sentence is passed through a bi-attention mechanism [3] which outputs interdependent vectors. This is followed by an integration layer using separate bidirectional LSTM units on the concatenation of the attention layer outputs. The outputs are then passed through a pooling layer along the time axis followed by a three layer batch normalized maxout network which takes the two separate pool layer outputs and provides a prediction on the classification.

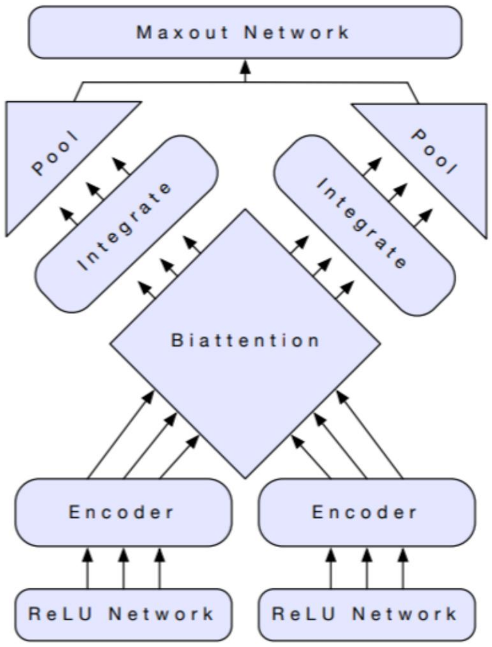


Figure 1: Sample figure caption.

3 Approach

3.1 Baseline

The baseline model consists of an embedding layer, a single layer LSTM with 50 units, and a dense layer with ReLU activation before a softmax layer which outputs the predictions. The embedding layer provides the word vectors for the words in a sentence by performing a lookup from the word embedding matrix.

3.2 Modified Bi-attentive Classification Network

Since our problem does not involve one AND two sentence classification tasks, we plan on using an architecture heavily inspired by the BCN, but not exactly it.

The model described by McCann et al [2] was modified to have a single sentence input, with the biattention mechanism replaced by self-matching attention, as seen in Figure ???.

The word embedding for a batch of sentences is passed through an initial ReLU network (10 units) and through a bidirectional LSTM encoder (50 units). The attention layer takes in a single sentence input, and follows implementation of self-matching attention from R-Net [5] with the attention scores calculated without a decoder vector.

$$s_j^t = \mathrm{v}^{\mathrm{T}} \tanh(W_v^P v_j^P + W_v^P v_t^P)$$
$$a_i^t = \exp(s_i^t) / \Sigma_{j=1}^n \exp(s_j^t)$$
$$c_t = \Sigma_{i=1}^n a_i^t v_i^P$$

Figure 2: Self-matching Attention Equations

The softmax of the attention scores is then multiplied element-wise by the output of the LSTM encoder layer. The attention layer output is then passed through a bidirectional LSTM (100 units). The output is then passed through a layer that performs both max and average pooling, followed by three dense layers (with 50, 50, and 25 units, respectively) with dropout layers in between these layers during training.

A cross entropy loss was first used, and then modified to a weighted cross entropy loss to account for the skew in the distribution of labels in the dataset. The Adam optimizer is used during gradient descent with the default hyper parameters.

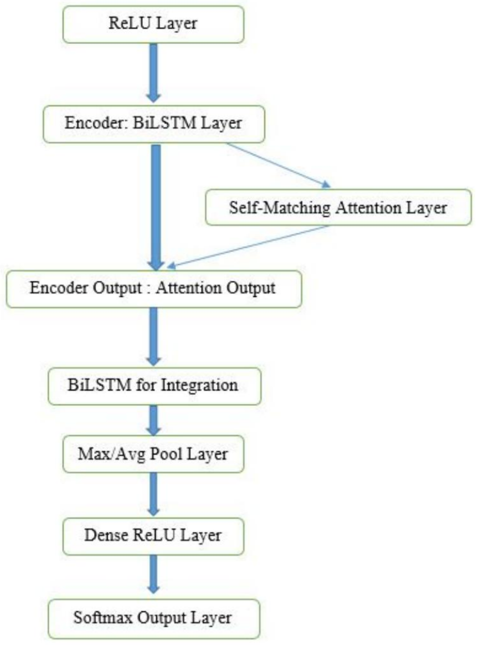
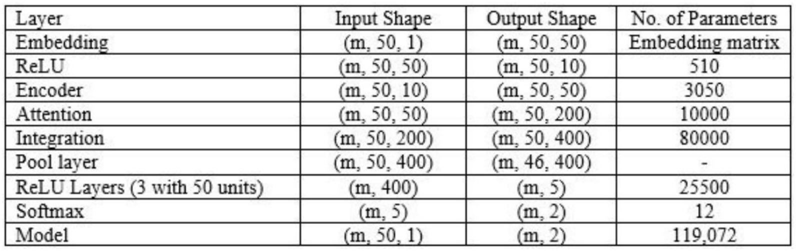


Figure 3: Modified Bi-attentive Classification Network Architecture for predicting funny Yelp Reviews

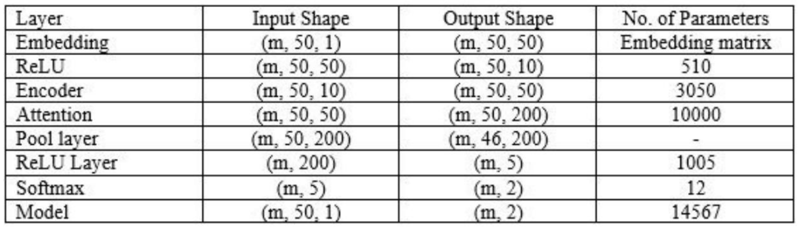


Layer	Input Shape	Output Shape	No. of Parameters
Embedding	(m, 50, 1)	(m, 50, 50)	Embedding matrix
ReLU	(m, 50, 50)	(m, 50, 10)	510
Encoder	(m, 50, 10)	(m, 50, 50)	3050
Attention	(m, 50, 50)	(m, 50, 200)	10000
Integration	(m, 50, 200)	(m, 50, 400)	80000
Pool layer	(m, 50, 400)	(m, 46, 400)	-
ReLU Layers (3 with 50 units)	(m, 400)	(m, 5)	25500
Softmax	(m, 5)	(m, 2)	12
Model	(m, 50, 1)	(m, 2)	119,072

Figure 4: Modified Bi-attentive Classification Network architecture's Number of Parameters – This will become important later due to overfitting issues.

3.3 Final Architecture

The implementation of the model was modified to overcome overfitting issues. The bidirectional LSTM layer after the attention layer was removed completely, and the number of dense layers at the end of the graph were reduced to 2, one with 5 units and the last one producing the softmax prediction. The weighted cross entropy loss is obtained from the labels and predictions along with an Adam optimizer for gradient descent.



Layer	Input Shape	Output Shape	No. of Parameters
Embedding	(m, 50, 1)	(m, 50, 50)	Embedding matrix
ReLU	(m, 50, 50)	(m, 50, 10)	510
Encoder	(m, 50, 10)	(m, 50, 50)	3050
Attention	(m, 50, 50)	(m, 50, 200)	10000
Pool layer	(m, 50, 200)	(m, 46, 200)	-
ReLU Layer	(m, 200)	(m, 5)	1005
Softmax	(m, 5)	(m, 2)	12
Model	(m, 50, 1)	(m, 2)	14567

Figure 5: Final Model used for classifying funny Yelp Reviews's Number of Parameters

4 Experiments

4.1 Dataset and Evaluation Metrics

Yelp provides a set of review, business and user data [2]. This dataset contains about 5 million reviews. In each review, there is the review's author's user id, business id, star rating, date written, number of funny votes, number of cool votes, and text of the review. For this project, we only analyzed the text, so our input only includes the (processed) text. Due to memory issues in the more complex models due to the increasingly large number of parameters, we decided to limit the longest considered review to have 50 words.

The variable we are predicting is whether a review is funny or not based on whether it received any funny votes. The dataset provides the number of funny votes each review has received from the user. With our constrained data set, 11 percent of reviews are considered funny.

The data has been divided into train and dev sets along with a test set of the data that will be kept aside for testing at the end of the project. The train set contains about 1.3 million reviews, the dev set contains 70,000 reviews, and the test set contains the 70,000 reviews.

This set of reviews contains about 70,000 unique words in its corpus. Each review will be fed as a set of word embeddings with a dimension of 50. All models will have the pre-trained word embeddings from GLoVe's 6 billion vocabulary set embeddings of 50 dimensions.

For fairness and comparability, all of the models are trained and evaluated upon this set of data.

For evaluation metrics, we used primarily F1 scores, although we did keep track of precision, recall, and accuracy. Precision and recall were tracked to both help calculate the F1 score and also figure out how to approach improving the model(s).

In addition, we used the F1 score rather than the accuracy because the large imbalance in funny to not-funny reviews made accuracy a very biased metric. For instance, a model that just predicts 0s could obtain 85 percent in accuracy, even though the model did not learning anything about predicting funny reviews.

Now, we will go into depth about our experiments, with a summary of the experiment results at the end of the experiments section at Figure 3.

4.2 Baseline

Our baseline, (discussed above in approach) features an LSTM cell. It was run for 10 epochs, since each epoch would take between 30 minute to an hour, even when running on Azure with a batch size of 2096. In addition, this baseline featured 30 percent dropout to discourage overfitting, had 50 LSTM units, and had a dropout rate rate of 0.03. The learning rate was also 0.001.

Ultimately, the baseline model performed very poorly with a F1 score of 0, as precision and recall were all 0. However, accuracy was 88.4 percent on both the dev and test sets. Consequently, our baseline model - even though it featured an LSTM - learned to predict everything as a 0.

4.3 Modified BCN

After creating and running the baseline, we implemented and experimented upon our modified BCN model. At the beginning, we used the same hyper-parameters as the baseline with the exception of batch size (i.e., 30 percent dropout rate, 10 epochs, a learning rate of 0.001, 50 LSTM units). Due to the size of the model, the largest batch size we utilized without crashes was 64. In addition, we used a unweighted loss function.

This modified BCN gave us the same results as the baseline: 0 for an F1 score, precision, and recall along with an 88.4 percent accuracy. Clearly, the issue was the unequal distribution of weights rather than the (lack of) complexity of the model.

4.3.1 Modifying Loss

Therefore, we added weights to the loss, with the first attempt making each funny review worth 4 not funny ones. To isolate the impact of the changes, all other hyper-parameters were kept the same (i.e., 30 percent dropout, 10 epochs, a learning rate of 0.001, and 50 LSTM units).

This modified model did significantly better on the train set, with a F1 score of 38, recall of 80 percent, precision of 25 percent, and accuracy of 70 percent. However, the model was overfitting, as the dev set had a F1 score of about 20, with a recall of 56 percent, accuracy of 58 percent, and precision of 15 percent.

4.3.2 Applying Additional Regularization

After trying to change the dropout without any difference, we decided to add additional regularization through adding L2 regularization to the loss function. All the other parameters stayed the same to isolate the changes (i.e., 30 percent dropout, 10 epochs, a learning rate of 0.001, and 50 LSTM units).

However, this addition caused the model to underfit. The F1 score was 20, but precision was 11.5 percent and recall was 100 percent along with an accuracy of 11.5 percent. Therefore, our model was now predicting everything as 1s.

In order to see if L2 regularization by itself was causing the underfitting, we lowered the dropout rate to 0 percent while maintaining the other parameters. Ultimately, we got the same result, as before.

This result meant that the model was too complex and had many, many parameters with weights close to 0, causing the model to underfit when the weights were incentivized to become even closer to 0 (with L2 regularization). Therefore, the solution was to decrease the complexity of the model.

4.4 Final Architecture

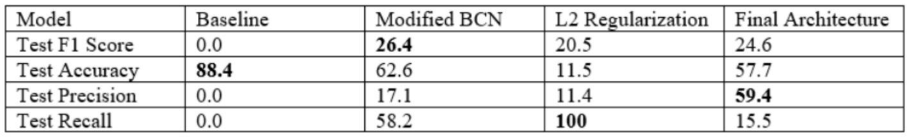
After simplifying our modified BCN model, we reached our final architecture. Under this architecture, we did some hyper-parameter search and obtained our best F1 score: 25.

In this score, we had an accuracy of 60.7 percent, precision of 15.3 percent, and recall of 61.2 percent.

4.5 Overall Results

The summary of our results on the test set can be seen in Figure 6. From the table, we can see that using the modified BCN (with the loss being the modified one), obtained the best F1 score.

Going through the test set predictions, we see that the model tends to label reviews that have big shifts in topics discussed as funny. For instance, the model correctly labeled ”I can’t wait to get to my bed from SOMA. It’s the most comfortable bed I’ve ever laid on. The knowledge and information given was outstanding, and it’s a no brainer if you want to go,” as funny. However, reviews often do switch between topics, which caused a lot of errors. For instance, the model incorrectly labeled this review as funny, ”Go here, you won’t regret it. The bartenders are so friendly and have tricks up their sleeves. Cam here for a cold one and wound up staying for hours just because we had so much



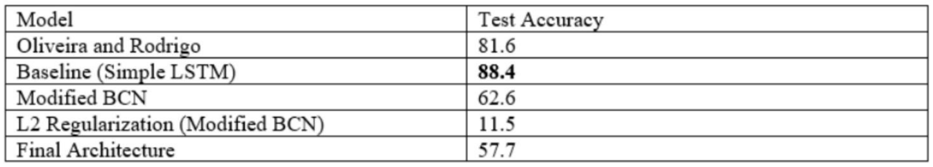
Model	Baseline	Modified BCN	L2 Regularization	Final Architecture
Test F1 Score	0.0	**26.4**	20.5	24.6
Test Accuracy	**88.4**	62.6	11.5	57.7
Test Precision	0.0	17.1	11.4	**59.4**
Test Recall	0.0	58.2	**100**	15.5

Figure 6: Test Set Evaluation Results for the Experimental Models

fun. Good find," as the model most likely thought that "cold one" was referring to being cold rather than about beer, hence making this review a very jarring one in terms of topics covered.

In addition, the labelling of funny reviews is quite subjective, causing predicting whether or not a review is funny to become an even more complex task. For instance, the review "I haven't had my hair done in over a year and was scared. Jonathan was amazing and knew exactly what I wanted even thought I didn't really know what was best. My hair is amazing - natural balayage," was considered as a funny review, even though most wouldn't consider this as humourous. Therefore, this arbitrary nature makes the task even harder.

When comparing to other papers, no other paper used the exact Yelp data set to predict humor, with the exception of Oliveira and Rodrigo, except that they only used accuracy as a metric. Comparing on the metric of accuracy, our baseline model (i.e., the LSTM) does the best with an accuracy of 88.4 percent, even though the model holistically does not do well since it only outputs 0s. Therefore, accuracy is a flawed metric and should only be one of many, not the only, metric in evaluating models. It should also be noted that the results obtained in Oliveira and Rodrigo's experiments were obtained from a class balanced distribution of the dataset, which is different from the dataset used, which included a disproportionately large number of reviews that are not funny.



Model	Test Accuracy
Oliveira and Rodrigo	81.6
Baseline (Simple LSTM)	**88.4**
Modified BCN	62.6
L2 Regularization (Modified BCN)	11.5
Final Architecture	57.7

Figure 7: Comparison of Model Accuracy

5 Conclusion

The experiments conducted during the course of the project produce a final F1 score of 26.4 (P: 17.1, R: 58.2), which improves upon the baseline models F1 score of 0.0 (due to P:0 and R:0). The low F1 score indicates that the model has difficulty predicting the funny reviews. The mismatch between the cross entropy loss function for which the model is optimized and the F1 score which is our primary optimizing metric, combined with experimental constraints, causes difficulties in finding best model for the F1 score.

The model architecture could be improved by experimenting with other forms of attention or including convolution layers. The results could be improved with better data or a different neural architecture. In addition, the data provided to the model could be improved by including pre-trained word embeddings with larger dimensions, including word vectors trained on a corpus including humorous content, relevant features from the remaining data provided in the Yelp dataset such as restaurant and user information along with each review. The task of identifying humor within written text is an interesting challenge in the field of natural language processing with an opportunity for improvement by testing different architectures with better datasets for humor-related tasks.

6 References

[1] Yelp.com. (2018). Yelp Dataset. [online] URL: `https://www.yelp.com/dataset/challenge`.

[2] B. McCann, J. Bradbury, C. Xiong, R. Socher. Learning in Translation: Contextualized Word Vectors. 17th March 2018. URL: `https://arxiv.org/pdf/1708.00107.pdf`.

[3] D. Bertero, Pascale Fung. A Long Short Term Memory Framework for Predicting Humor in Dialogues Hong Kong University of Science and Technology. 17th March 2018. URL: http://www.aclweb.org/anthology/N16-1016

[4] L. Oliveira, A. L. Rodrigo. Humor Detection in Yelp Reviews Stanford University CS224D Project. 17th March 2018. URL: http://web.stanford.edu/class/cs224d/reports/OliveiraLuke.pdf

[5] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. ICLR, 2017. URL: `https://arxiv.org/pdf/1611.01603.pdf`.