Assessing the Funniness of Edited News Headlines: Task 1

Hanna Behnke
Imperial College London
hsb20@ic.ac.uk

Sneha Naik
Imperial College London
sn914@ic.ac.uk

Jamie Salter
Imperial College London
jas20@ic.ac.uk

1 Introduction

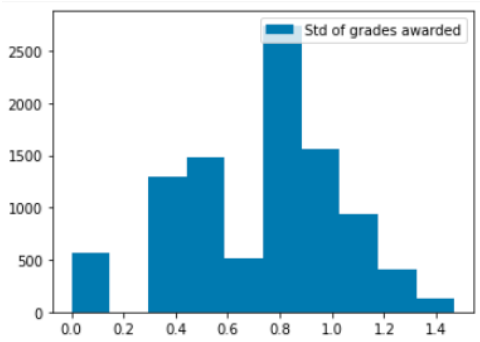
In this report, we discuss our approach to predicting the funniness scores of modified news headlines. First, we outline our data exploration process which serves the purpose of understanding the problem and deciding on suitable preprocessing steps. Next, we present our experiments, starting with those that build upon pretrained language models and continuing with approaches that rely on self-trained language representations. Both sections include a description of the pre-processing, model design choices and training process, followed by a discussion of the obtained results.

2 Data Exploration

The dataset consists of 12,071 headlines of which 20% are used as an unseen validation set for hyperparameter tuning. Another 3,024 headlines serve as the held-out test set. Per headline, the dataset contains information about the original wording, the edited word, the scores given per voter (0-3) as well as the mean score.

We analysed the grades per headline from various perspectives, seeking to understand how the scores were assigned, what makes a headline funny and if there are any outliers that should be excluded form the training set. The mean grade roughly follows a normal distribution, shifted towards the lower end of the grading spectrum. From reading selected headlines, we noted that in particularly funny headlines, the edited word seemed more surprising in the context of the original sentence which we will explore in further detail in section 4.2.1. Further, we found that "Trump" is mentioned frequently in funny headlines, however, a deeper analysis showed that the correlation with the mean grade is only marginal.

Figure 1: Standard deviation of grades per headline



All headlines were judged by at least 5 people, however, there appeared to be a strong disagreement about the funniness of some of them. Figure 1 shows the standard deviation of the votes per headline. We decided to exclude all headlines with a deviation larger than 1.2, assuming that they would hardly help the model to learn an unambiguous representation of humor.

3 Approach 1 (Pre-trained Language Representations)

3.1 Architecture details

Three architectures were considered. Firstly, we fed GloVe embeddings (100-dim) [1] into a BiLSTM to produce a context vector that serves as input to a feed-forward neural network (FFNN) used for regression. After that, we compared two different language models, BERT and Electra, each followed by a FFNN for regression [2]. Both BERT and Electra have been trained on over 3 billion words, but the training regimes are significantly different. BERT uses Masked Language Modelling (MLM), masking out 15% of words during training, and Next Sentence Prediction, whereas Electra uses a poor performing generator to replace words in a sentence that a discriminator (our pre-trained model) has to classify as original or edited. Electra classifies all words in the sentence, whereas MLM only predicts each masked word. This gives Electra a performance advantage, so we would expect Electra to be our best model.

3.2 Input data and preprocessing

Different pre-processing steps were considered for the BiLSTM experiments versus those using Electra and BERT. For BiLSTMs we firstly identified and removed out of vocab words. We then removed any hyphens from these words and checked again so as not to break down words unnecessarily. For BERT and Electra, we utilised the Bert tokenizer and Electra tokenizer from HuggingFace for compatibility. In all experiments, punctuation was not removed to prevent losing any context.

3.3 Training Procedure

To find a suitable BiLSTM architecture, we ran four experiments which build on each other. Firstly, we investigated the impact of varying the learning rate. After the best learning rate was identified, we experimented with different architecture settings, using 1-2 LSTM layers (dim =

50) and 0-2 linear layers (dim = 20, dropout = 50%). We chose ReLU as the activation function and Sigmoid scaled by 3 as output activation. Using the best-performing architecture, we investigated the impact of different input features, and finally, we un-froze the embedding layer and experimented with its learning rate. All experiments were run for 10 epochs with a batch size of 16, using AdamW as optimiser and RMSE as loss function. To find the best parameters, we used 2-fold cross-validation in a grid search. The training dataset was split into 80% training set, 20% validation set.

When investigating input features, our default concatenation approach was to put the original and edited sentences through the same LSTM separately, and then concatenate the result into a vector of twice the length before entering the FFNN. We hypothesised that this way, the LSTM could learn a 'funniness' representation for each sentence and that the FFNN could model the "relative funniness" of the edit and the original. Apart from concatenation, we further tried to subtract or multiply the representations.

We then experimented with BERT and Electra using 3-fold cross-validation (rather than 2-fold due to increased variability of results). Training was conducted with non-frozen embeddings with a learning rate warm up and cool down schedule over 2-3 epochs. The FFNN was kept consistent with the best configuration found in the BiLSTM experiments. After hyper-parameter tuning, we investigated the impact of feeding each sentence through the transformer separately and then concatenating, versus passing through both simultaneously, separated by a [SEP] token. For Electra, we also varied the training epochs to see the impact on results.

Table 1 shows the performance of the models with different hyperparameters.

3.4 Results and Discussion

Firstly, from model IDs 4-9, we discovered the achieved RMSE was relatively insensitive to the number of stacked LSTMs and deeper classification layers. The best model (ID 5) shows only a 2% improvement over the worst (ID 4) in this experiment. However, we see that the model tends to predict a narrow distribution around the mean, indicating that the training data is the limiting factor rather than the capacity of the model. However, we note that only 2-fold cross-validation has been used, therefore the results are not noise free.

Secondly, model IDs 10 to 13 show the results of varying how the original and edited sentences were combined. Although we expected a benefit from concatenation, we observed that the best model (RMSE of 0.539) used the edited sentence only as input. This indicates that the benefit of additional information from the concatenated representation is outweighed by the added complexity.

Thirdly, unfreezing the embedding layer and varying it's learning rate in model IDs 14-16 caused severe overfitting due to smaller learning rates. This is expected, since the model had 63M parameters and we had only 12k headlines.

Overall, the best BiLSTM model was model ID 13, achieving an RMSE of 0.5393 on the development set. The model used a frozen embedding layer, a learning rate of 0.001 (from ID 1) and a single LSTM and single FFNN layer (from ID 5) and had only the edited sentence as input.

When comparing BERT and Electra with varying input representations, we see that the performance difference between the two fluctuated. BERT was 1.3% better than Electra when using edited sentence only (IDs 19 & 22). Electra performed 3.4% better when using the combined representation with a SEP token (IDs 18 & 21). This indicates that Electra makes better use of the additional context provided. Notably, for both BERT and Electra, training the embedding layer yielded better performance than freezing it, unlike for the BiLSTM. This is likely due to the huge size of the language models relative to single layer GloVe so our additional training has less impact. We also note that Electra was particularly sensitive to the number of epochs trained for on training set performance due to the steep warm-up / cool down learning rate schedule.

Overall, the best model for Approach 1 was Electra with a single FFNN layer for regression (ID 21). Testing the model on the competition test set yielded an RMSE of 0.5136 after 2 epochs [6].

4 Approach 2 (Manually Trained Language Representations)

4.1 Preprocessing

We conducted extensive pre-processing due to the limited corpus available for training our word representations. We lowercased all words, removed punctuation and stopwords, applied stemming (PorterStemmer) and added start and end to-

ID	Model	Attributes changed	T loss	V loss	ID	Model	Attributes changed	T loss	V loss
1	BiLSTM	Learning rate: 0.001	0.5283	**0.5466**	14	BiLSTM	Embedding learning rate: 0	0.5530	**0.5439**
2	BiLSTM	Learning rate: 0.0001	0.5751	0.5695	15	BiLSTM	Embedding learning rate: 0.0005	0.4700	0.5506
3	BiLSTM	Learning rate: 0.00005	0.6092	0.5816	16	BiLSTM	Embedding learning rate: 0.00005	0.5414	0.5559
4	BiLSTM	LSTM layers: 1 & FFNN layers: 0	0.5463	0.5516	17	BERT	Combination: edited 'concat' original after	0.5343	0.5415
5	BiLSTM	LSTM layers: 1 & FFNN layers: 1	0.5517	**0.5409**	18	BERT	Combination: edited [SEP] original before	0.5648	0.5649
6	BiLSTM	LSTM layers: 1 & FFNN layers: 2	0.5582	0.5634	19	BERT	Combination: edited only	0.4955	**0.5409**
7	BiLSTM	LSTM layers: 2 & FFNN layers: 0	0.5467	0.5493	20	Electra	Combination: edited 'concat' original after	0.5704	0.5724
8	BiLSTM	LSTM layers: 2 & FFNN layers: 1	0.5529	0.5480	21	Electra	Combination: edited [SEP] original before	0.5024	**0.5236**
9	BiLSTM	LSTM layers: 2 & FFNN layers: 2	0.5578	0.5487	22	Electra	Combination: edited only	0.5304	0.5446
10	BiLSTM	Combination: edited x original	0.5583	0.5528	23	Electra	2 epochs	0.5024	0.5236
11	BiLSTM	Combination: edited - original	0.5559	0.5539	24	Electra	3 epochs	0.3884	**0.5230**
12	BiLSTM	Combination: edited 'concat' original	0.5514	0.5464					
13	BiLSTM	Combination: edited only	0.5558	**0.5393**					

Table 1: Minimum training RMSE (T loss) and validation RMSE (V loss) for different hyperparameters and models

ID	Input	Embedding	Model	Dev RMSE	Dev MSE	Key Parameters
1	Edit word	Original	TfIDF baseline	0.60	0.36	unigram only
2	Edit word - Original word	W2V original	SGDRegressor	0.5764	0.3323	learning_rate: 'adaptive', penalty: 'l1'
3	Edit word - Original word	W2V original	SVR	0.5840	0.3411	kernel: linear (better than rbf, polynomial)
4	Edit word - Original word	W2V original	ExtraTreesRegressor	0.5849	0.3421	n_estimators: 80 (searched 10-100)
5	Edit word - Original word	W2V augmented	SGDRegressor	0.5900	0.3481	learning_rate: 'adaptive', penalty: 'l1'.
6	Edit word - Original word	W2V augmented	SGDRegressor	0.5921	0.3507	as in ID 5, with "UNK"token
7	Edit word - Original word	W2V augmented	MLPRegressor	0.6186	0.3826	hidden dim (100,100), ReLU, L2 reg 0.001
8	Edit word - Original word	W2V augmented	ExtraTreesRegressor	0.6062	0.3674	n_estimators: 100
9	Edited headline	W2V augmented	BiLSTM	0.5776	0.3337	config from approach 1 (ID 13, Table 1)
10	Edit word - Original word	FastText augmented	SGDRegressor	0.5913	0.3496	learning_rate: 'adaptive', penalty: 'l1'

Table 2: Minimum RMSE and MSE on the development set for selected experiments conducted. Key parameters identified from grid search. Where not specified Sklearn default parameters were used. Unless noted, headlines with unseen words were skipped.

kens to the sentences (for use in sequential models). We discovered that 25% of the words in the test set are not included in the original training set, so we experimented with different techniques to handle rare and unseen words (subword methods / unknown tokens).

4.2 Learning Word Representations

After initial experiments with sparse Bag-of-Words representations, we investigated three embedding approaches to account for context: Word2Vec using the original corpus, Word2Vec using a larger training corpus and FastText embeddings. All three reduce the dimensionality of the input data to 100 dims, in line with Approach 1. We chose a window-size of 2 as larger window-sizes led to significant overfitting in the regression task. For training, we used Gensim [3], an efficient alternative to our own implementation.

When training on the original training corpus only, we discovered that many words, most crucially edited words, only appear once in the dataset. Consequently, the learned word representations might be biased or even based on a context where the (edited) word does not naturally occur. To tackle this issue, we decided to train a second W2V model on a larger training corpus, including "UNK" tokens for words that would still appear only once. We added more than 209k news headlines with a broad range of topics and a similar time-frame utilising the "Funlines" dataset [4] and the "Kaggle News Category" dataset [5] (the latter to ameliorate our need for 'normal' contexts).

Using Gensim's 'most similar" function which returns similar words for a given token based on the distance of the embeddings, we showed that the augmented model outperformed the smaller version by far. For 'Merkel', the large model returned the names of other politicians while the small model returned tokens like "playboy", "villain" and "joint", which were clearly learned based on the funny edits. Using Gensim's similarity scores, we further showed that an edited word that is less likely to occur in the original context is more surprising, and therefore more funny: the mean similarity between the original and edited word vectors for headlines with a high funniness score of more than 2.2 was **half** that of boring headlines with a funniness of less than 0.1 (similarity of 0.06 and 0.12, respectively). This confirmed our hypothesis from Section 2.

As another approach to handling unseen words, we experimented with byte-pair encoding ahead of Word2Vec, however, applying enough merges to generate feasible tokens was intractable due to computational complexity and limited vocabulary size. Therefore, we used FastText, an embedding method which learns sub-word embeddings.

4.2.1 Training procedure

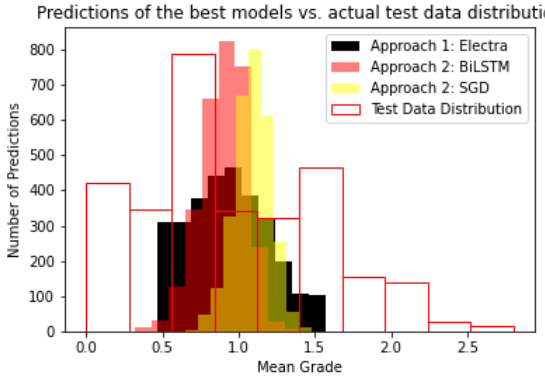
To find a suitable architecture we ran a grid search over hyperparameters using an 80% train and 20% validation split with 2-fold cross-validation on four non-sequential Sklearn models (Extra Trees Regressor, SGD Regressor, MLP Regressor, Support Vector Regressor (SVR)). We conducted three key experiments. Firstly, we examined the difference in RMSE from training our Word2Vec embeddings on the original versus the augmented corpus. Secondly, we analysed the impact on RMSE of replacing unseen words at test time with the unknown token versus removing the out-of-vocab words. Finally, we investigated whether the use of FastText embeddings trained on the augmented dataset would obviate the need for unknown tokens and give a better RMSE.

Further, we trained the best-performaing BiLSTM architecture from Approach 1 using our learned Word2Vec embeddings instead of Glove to compare the relative quality of the two embedding types while keeping the remaining factors constant, which not only account for local, but also, global context.

Regarding the input features for our non-sequential models, we decided to use the edit word minus the original word embedding. This exploited the dissimilarity hypothesis proposed in Section 4.2. For the BiLSTM, we used the edited headline's word vectors as input, as this configuration yielded the best results in Approach 1.

The results of our experiments, and optimal hyperparameters from our grid searches are detailed in Table 2 and the learned distributions are shown in Figure 2. We analyse our findings in Section 4.3.

Figure 2: Distribution of the predictions of the 3 best models on test set. Final RMSE: SGD 0.584, BiLSTM (W2V) 0.5776, Electra 0.5136



4.3 Results and Discussion

The best non-sequential model was the SGD regressor, however, it also gave the narrowest distribution of predictions. The more flexible MLPRegressor gave a wider range of predictions but worse RMSE. When replacing unseen words with "UNK" rather than skipping the headline, the output results were 0.7% worse across models (Table 2, IDs 5 vs 6). This suggests that the information encoded in the "UNK" embedding is not helpful for the downstream task (words "most similar" to unknown tokens include "glenda", "shulnik"). The performance of FastText embeddings with the SGD regressor was slightly worse than that of the Word2Vec embeddings. This could be due to the small training corpus limiting the ability of FastText to learn morphological details of words.

The best performing model was the BiLSTM which highlights the importance of context to the humour of the sentence. Compared to the Glove embeddings, however, using our Word2Vec augmented embeddings resulted in 6.2% worse performance. This result is unsurprising, given that Glove accounts for global context and is trained on $\sim$ 6 billion tokens (not just $\sim$ 2 million tokens).

We were surprised to observe a lack of performance improvement between W2V original and W2V augmented embeddings given the significant difference in quality discussed in Section 4.2. We attribute this to the difficulty of the regression task: from Figure 2, we see that all of our models, even the best model from Approach 1, were unable to represent the true distribution and tend to predict the mean of the data.

5 Conclusion

Overall, we found that a good context representation improves the regression results. The architectures based on pre-trained LMs clearly outperformed all experiments from Approach 2. However, the tendency of the models to generate predictions close to the mean indicates how difficult it is to learn a relationship between the edited headlines and associated funniness. After all, humor is highly subjective: as a group, we attempted to predict the funniness score of 50 headlines each and achieved an RMSE of only 0.96 which is even worse than the baseline. Given that the headlines were graded by different people who have a different perception of funniness, we can hardly expect our ML models to detect clear patterns in the data.

A References

[1] GloVe: Global Vectors for Word Representation. Jeffrey Pennington, Richard Socher, Christopher D. Manning, 2014. https://nlp.stanford.edu/projects/glove/

[2] Transformers — transformers 4.3.0 documentation https://huggingface.co/transformers/

[3] Gensim. Radim Řehůřek. https://radimrehurek.com/gensim/

[4] FunLines Humour Dataset. Nabil Hossain, John Krumm, Tanvir Sajed and Henry Kautz, 2019. https://www.cs.rochester.edu/u/nhossain/funlines.html

[5] News Category Dataset. https://kaggle.com/rmisra/news-category-dataset

[6] Submitted to CodaLab for verification https://competitions.codalab.org/competitions/20970#results