70016 Natural Language Processing Coursework Report

Arthur Cauchetier
Department of Computing,
Imperial College London,
London SW7 2AZ, UK
ac920@ic.ac.uk

Tien-Chun Wu
Department of Computing,
Imperial College London,
London SW7 2AZ, UK
tw2220@ic.ac.uk

Marek Topolewski
Department of Computing,
Imperial College London,
London SW7 2AZ, UK
mt820@ic.ac.uk

Abstract

We aim to quantify humor by developing machine learning models predicting which of two edited news headlines is funnier. Several approaches are proposed and evaluated. The performance is assessed on a test set of 2961 samples for which the fine-tuned BERT model and the not-pretrained BiLSTM with attention achieved 61% and 54% accuracy respectively.

1 Introduction

Humor is an inherently hard feature to measure. It is not only difficult to detect for a human but even more so for a computer intelligence. Main contributor to this fact is the subjectivity of funniness which can be accredited to several factors: familiarity, political views, global context, mood etc.

The challenge is formalised in a competitive setting introduced in *SemEval-2020 Task 7* (1). This project focuses on the second sub-task of the competition: given an original headline and two *atomic* edits, decide which of the modifications make the new sentence funnier.

Two contrasting approaches were followed to solve the problem:

1. **Pretrained** - utilise extensively pretrained models such as BERT (2) and RoBERTa (3). Subsequently fine-tune the model for a specific task.
2. **Not pretrained** - employ a system that can be trained from the ground up, both in terms of the embeddings and the classifier.

The implementations of these two approaches can be found on this GitHub repository.

2 Dataset

The primary data source utilised in this project is the Humicroedit dataset (4). Each entry consists of a pair of headlines, associated *funniness* grades and a label as the index of the more humorous edit.

To augment the dataset, another source is used: FunLines (5). The structure of the data is similar, however, FunLines introduces more versatile edits but contains fewer samples.

Data is divided into two disjoint sets: training (and development) and test. The training set is further split in an 80:20 ratio to obtain a validation set used for model evaluation, hyperparameter search, early stopping to avoid overfitting. The test set is completely isolated to provide an unbiased assessment of model's final performance.

An important observation, is that entries with 0 labels are ignored in the competition (1), hence, the multi-label classification problem can be reduced into a binary one. Because equally humorous pairs are rare (around 10% for both datasets), such cases can be ignored without the risk of lowering the dataset quality substantially.

To further increase the number of samples for training, a *random flip* is implemented. Exclusively in classification (see subsection 3.1), this transformation increases the dataset size by a factor of 2 and promotes symmetry, i.e. if 0 is assigned for input sentences (A, B), then label 1 should be produced for (B, A).

3 Approach 1

BERT (2), has emerged as the state-of-the-art for a wide variety of NLP tasks. The bidirectionnal architecture, the use of self-attention and the deep architecture enables it to achievegood performance regardless of the domain. (6).

Although training BERT is expensive (7), libraries like *HuggingFace Transformers* (6) provide numerous pretrained transformer-based models which greatly reduces development effort.

3.1 Architecture

Firstly, we considered taking advantage of the ability of BERT to encode multiple sentences jointly with the `<sep>` token. However, such a model would not be symmetric: computations applied to sentence A are different than to B. Considering that the sentences could be swapped, this design was suboptimal. Consequently, we pass each sentence individually which preserves symmetry and avoid computational redundancy.

Although BERT generates powerful embeddings, its outputs require further processing. To obtain the predictions, a learnable *head* is implemented and two approaches are considered:

1. **Regression head** - train a linear layer to predict the mean grade of each embedding, use *argmax* to label the higher score.
2. **Classification head** - the two output embeddings are concatenated and fed to a classifier.

An important modification to the base BERT model was the span representation which yields a fixed-length encoding (8). A span pair is constructed by splitting the sequence of embedding tokens such that the first element corresponds to the end index of tokens preceding the edit, and the second to the start index of those following it (9).

Apart from employing vanilla BERT, the RoBERTa variant was also tested. This model expands the pretraining by ignoring the NSP objective, adding dynamic masking and increasing: dataset size, training time, batch size (3). The result of these adaptions is a performance improvement at the cost of pretraining time (10).

3.2 Features

Three sentences, that differ only by the edited subsequence, are considered per edit: original, edited, and masked. Each headline is subsequently tokenized by first wrapping the distinct strings with `<sep>` tokens, and later the vectorized using the pretrained tokenizers. Depending on the BERT configuration, different tokenizer is required: base BERT uses WordPiece (11), while RoBERTa relies on BPE (12).

Subsequently, a headline is propagated through the transformer to obtain the embedding. Any two representations can be merge into a feature vector defined by a binary function f_i:

$$f_1(u,v) = [\, u, v, |u-v|, u*v \,]$$
$$f_2(u,v) = [\, u-v \,]$$

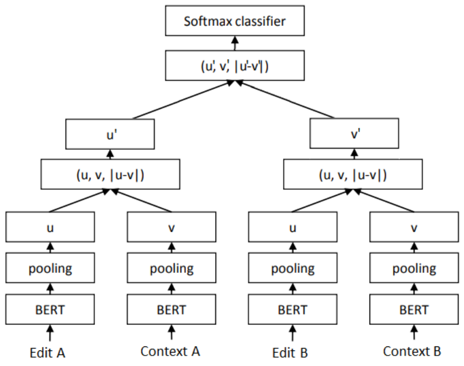

Figure 1: An adaptation of the SBERT architecture to incorporate the sentence context in edit classification.

where $|u-v|$ is the absolute element-wise difference, and $u*v$ is the element-wise product of embedding vectors u and v (13; 9).

If u and v are the two edited sentences, they can directly be used for classification, hence, for the model to make informed predictions, f_i cannot be commutative. f_1 encodes this asymmetry by stacking u and v but f_2 does so via signed element-wise difference. A regressor considers only a single headline at a time, therefore, an edited headline and its context (original or masked headline) can be leveraged to represent semantic divergence. The same principle is applied to classification but the two contextualised features must be reprojected into a single vector (see Figure 1).

3.3 Training

Loss function choice depends on the model head. For classification, the CE of label distributions is used. Because 0 labels are ignored, the softmax is replaced with the sigmoid function and the BCE is minimised instead. For regression, the MSE of mean grades is optimised.

We chose AdamW optimizer (Adam with weight decay fix), a common choice for BERT fine-tuning. Moreover, the literature encourages to use batch size of 32 or 64 and a learning rate between 10^{-5} and 10^{-4}. Consequently, we performed a hyperparameter research within this space and obtained the best results for a batch size of 32 and a learning rate of 2×10^{-5}.

To mitigate overfitting, early stopping is employed. The validation accuracy is an unbiased estimate of test performance, hence, if it increases then training is halted. Furthermore, gradient clipping is utilised to stabilise fine-tuning. Prior to each optimizer update, the gradients calculated in the backward pass are regularized using an L2 vector norm which prevents gradient explosion (14).

3.4 Evaluation

The best validation accuracy of 80% and test accuracy 61% was achieved by RoBERTa with a regression head and only edited headlines as inputs. While contextualisation did not decrease the accuracy, it greatly increased the training and inference time. The classification head although intuitively reasonable - the classifier compares extracted features instead of mean grades which should generalise better due to disparity in grades, did not outperform the simpler regression, furthermore, it was more prone to overfitting. There was no substantial improvement switching from BERT to RoBERTa, except for initially faster learning.

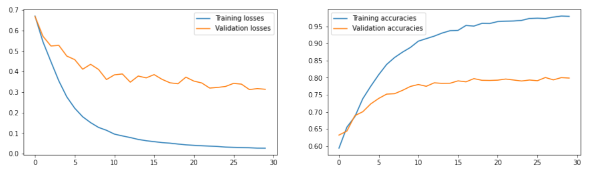

Figure 2: Training and validation MSE loss (left) and accuracy (right) for the regression head.

4 Approach 2

4.1 Architecture

Recursive neural networks (RNN) have been proven to be effective for machine translation and sentiment analysis tasks. LSTM families, in particular, resolve the vanishing gradient problem in RNN, making them powerful for modeling long-time dependencies. Bidirectional LSTM (BiLSTM) is an extension of LSTM which exploits both the past and future input features to achieve a thorough understanding of the sentences.

Our model comprises of an embedding layer, a BiLSTM layer, an attention layer, and a dense layer which learns the weights derived from 3 input features u, v, $|u - v|$ that utilise the BiLSTM with attention architecture (see Figure 3).

To understand the attention mechanism built upon BiLSTM layer, we represent its effect by the following equations : $c^{<t>} = \sum_{t'} \alpha^{<t,t'>} h^{<t'>}$ and $\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{t} \exp(e^{<t,t'>})}$, where $c^{<t>}$ represents the context vector to be fed into the subsequent dense layer. $h^{<t'>}$ is the output of hidden layer at t'-th layer. $\alpha^{<t,t'>}$ is the softmax weighting at t-th layer attributed to t'-th layer from the embedding $e^{<t,t'>}$. Attention layer is essentially

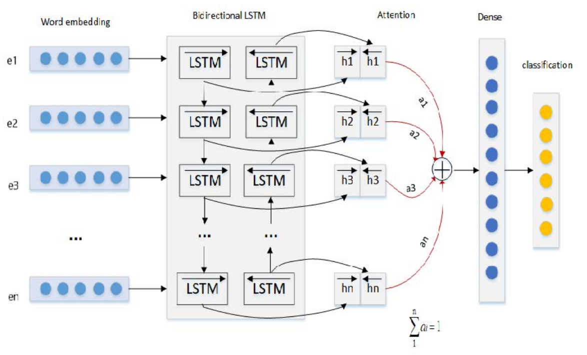

Figure 3: An example of BiLSTM Attention architecture. Adapted from (15).

a layer of neural network with softmax function. The importance of words is measured through the similarity between the i-th word and the word level context vector. Therefore, the method represents an efficient way to summarise the context of each sentence.

We then apply BiLSTM with attention on 3 input embedding vectors u, v, and $|u - v|$ (similarly to approach A), and concatenate the resultant representation with context to a dense layer to learn the weighting of each vector. Finally, we use cross-entropy loss to select the funniest vector.

4.2 Preprocessing and Embeddings

For text preprocessing, we removed the unnecessary punctuations (semicolon, colon) and stopwords using `nltk.corpus`. We discarded characters linked to social media (brackets, # and $\tilde{)}$ and removed the useless video and web links. We eliminated short headlings (less than 3 words) that may not convey sufficient meaning. We evaluated the new performance of these preprocessing and achieved a 2% improvement on validation set.

After preprocessing, we tokenise the headlines and build the vocabulary class with token for padding, start/end of headline, unknown word, and separator. Word embeddings were used instead of one-hot encoder (vocabulary too large) or bag of words (vectors too sparse).

A skip-gram model with a window size of 2 has been used to learn word embeddings. In order to reduce training time, negative sampling has been implemented by sampling 5 random words. Our first idea was to train this model on a large corpus, e.g MEANTIME corpus. However, we realised that it was too computationally expensive for such a large dataset. Consequently, we simply train the skip-gram model on the training headlines corpus.

4.3 Features

From empirical evaluations, we observe that the performance obtained with random embeddings or with embeddings trained with a skip-gram model on our corpus is comparable. As training a skip-gram model is very time consuming, we simply kept random embeddings for the initialisation of the BiLSTM model with attention.

4.4 Training

In the network, cross-entropy loss is implemented for binary classification task. Similarly to part 1, we used AdamW optimizer and gradient clipping to avoid gradient explosure. Dropout layers are incorporated into the BiLSTM to reduce overfitting.

We implement random search for the hyperparameters – learning rate, batch size, size of hidden dimension, and size of embedding dimension. Learning rate is sampled from a log-normal distribution between 10^{-5} and 10^{-3}; batch size is sampled from a random choice of values 8, 16, 32, 64, 128; the size of hidden dimension and the size of embedding dimension is sampled from a uniform distribution between 10 and 100. The following table shows the hyperparameter search results.

learn rate	# batch	hid dim	embed dim	val acc
1.13e-5	128	38	17	0.51
1.40e-5	32	54	80	0.51
1.67e-4	32	62	17	0.51
1.80e-4	64	24	88	0.53
2.04e-4	64	88	79	0.53
2.61e-4	128	10	63	0.51
7.09e-4	32	83	46	0.54
7.99e-4	64	52	58	0.51
9.55e-4	8	34	29	0.53

It appears that the dominant factors are learning rate and batch size. The optimal learning rate lies at the order of 10^{-4}. The best model is obtained with the following values: learning rate = 7×10^{-4}, batch size = 64, embedding dimension = 83, hidden dimension = 46.

4.5 Evaluation

Figure 4 shows the accuracy of training and validation sets evolving. The training accuracy reaches above 90% after 5 epochs, while the validation accuracy reaches the maximum (54%) at seventh epoch. Maximum performance saturates above 8 epochs. The training time is short which proves it as an efficient model. The performance of 54% accuracy on validation set is also above the benchmark for non pre-trained models.

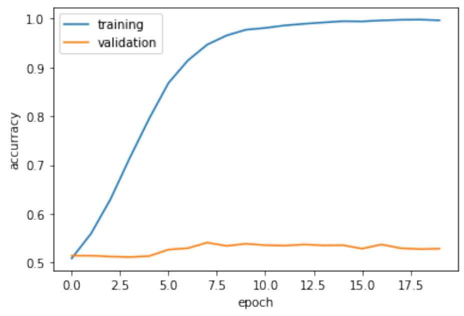

Figure 4: Accuracy of training and validation sets evolving over 20 epochs.

The incorporation of attention layer on top of BiLSTM model improves validation accuracy of ≈5% compared to vanilla BiLSTM. Whereas, the use of 3 input embedding vectors u, v, $|u - v|$ instead of 2 edited embedding vectors improves a further 5%.

5 Model comparison and discussion

The fine-tuned RoBERTa performed better than our Bi-LSTM with classification (61% compared to 54%). The promise of models such as RoBERTa was to be able to perform well on any task after little fine-tuning thanks to extensive pre-training. For this task, it seems that this aim has been reached. BERT models surpass specific pipelines by encoding a broader data scope at pre-training.

With RoBERTa, different approaches have been tried and revealed some surprising results. In particular, the regression head outperformed the classification one, despite not being directly optimised for this specific task. It could be due to the mean square error (linked to the degree of funniness of the sentence) carrying more information than the BCE used for classification. Moreover, we observed a large decrease in the performance going from validation to test set, probably due to differences in context and words observed.

The use of pre-trained BERT and RoBERTa proved beneficial despite longer to training time than Bi-LSTM with attention. This work allowed to understand the positive impact of pre-trained embeddings as it was not feasible to train our own embedding on the MEANTIME corpus.

Finally, 61% remains a weak result and even the best team performed 67% during the challenge. It reveals how challenging humor detection is and the vast improvements that need to be done to understand every nuance of human language.

References

[1] N. Hossain. (2021) Assessing the funniness of edited news headlines (SemEval-2020). [Online]. Available: https://competitions.codalab.org/competitions/20970

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[4] N. Hossain, J. Krumm, and M. Gamon, ""president vows to cut <taxes> hair": Dataset and analysis of creative text editing for humorous headlines," 2019.

[5] N. Hossain, J. Krumm, T. Sajed, and H. Kautz, "Stimulating creativity with funlines: A case study of humor generation in headlines," 2020.

[6] H. Transformers. (2021) Docs: BERT - overview. [Online]. Available: https://huggingface.co/transformers/model_doc/bert.html

[7] D. Antyukhov. (2021) Pre-training BERT from scratch with cloud tpu. [Online]. Available: https://towardsdatascience.com/pre-training-bert-from-scratch-with-cloud-tpu

[8] T. Kuribayashi, H. Ouchi, N. Inoue, P. Reisert, T. Miyoshi, J. Suzuki, and K. Inui, "An empirical study of span representations in argumentation structure parsing," Florence, Italy, pp. 4691–4698, Jul. 2019.

[9] S. Jin, Y. Yin, X. Tang, and T. Pedersen, "Duluth at semeval-2020 task 7: Using surprise as a key to unlock humorous headlines," 2020.

[10] S. Khan. (2019) BERT, RoBERTa, DistilBERT, XLNet — which one to use? [Online]. Available: https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use

[11] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[12] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2016.

[13] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[14] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2013.

[15] Q. Zhou and H. Wu, "NLP at IEST 2018: BiLSTM-attention and LSTM-attention via soft voting in emotion classification," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 189–194. [Online]. Available: https://www.aclweb.org/anthology/W18-6226