NLP Coursework

Vadim Becquet*
MSc Advanced Computing
vb720@ic.ac.uk

Joël Tang*
MSc Computing
(AI & ML)
jt620@ic.ac.uk

Victor Tempe*
MSc Computing
(AI & ML)
vt520@ic.ac.uk

Abstract

This document is the report of Vadim Becquet, Joël Tang and Victor Tempe for the NLP coursework challenge, related to the SemEval-2020 task: Assessing the Funniness of Edited News Headlines (SemEval-2020). The goal was to learn to assess the score of funniness (real value between 0 and 3) modified headlines, in a supervised manner. Many approaches are described in the notebook, using pre-trained embeddings or not. For each model description, a pre-processing method and results analysis are attached as well. Hyperparameters and designed choices are also explained.

* *Equal contribution.*

1 Introduction

Detecting funniness is a difficult task. Even among human beings, people from different cultures, backgrounds or with distinct personalities may disagree on what is funny or not. That is the case with the data set from SemEval-2020. We don't always consider the same edited sentence as funny even though an overall trend exists. Can Machine Leaning somehow detect humor? In this coursework, we tested several NLP and machine learning techniques to see if computers can. The task consisted in predicting in a supervised manner a score between 0 and 3 to assess how funny an edited news headline is. Although the data set is quite small, we obtained encouraging results and achieved to make some relevant predictions through various approaches.

2 Simple Conv+MLP with hybrid-learned embeddings

2.1 The model

The assumption is that we can predict the score of funniness from the relationships between the original sentence, the replaced word, and the replacing word. The relationship can be better extracted from embeddings of those components. We assume an embedding dimension of 100, and that we have $\vec{v}_{sentence}$, $\vec{v}_{replaced}$, $\vec{v}_{replacing}$. Then, assuming they are concatenated along the horizontal axis, we get a matrix of size 100x3 (one channel).

Architecture The matrix is fed to a convolutional layer with 8 output channels and a kernel of size (3, 1). Then, we have two fully connected layers, of output channels 8*100 $\rightarrow$ 16 and 16 $\rightarrow$ 1. All the non-linear activations are eLU (to prevent dying ReLUs).

The last layer activation was tested as ReLU or sigmoid. Both enforce the positivity, and the sigmoid (up to a rescale) can ensure that our score is capped to 3.0. In practice, ReLU model never predicted a value higher than 3.0. However, the sigmoid model provides a better RMSE.

Note that the shape of the convolutional kernel makes the assumption that we work on the embeddings representation element-wise: it is equivalent to put the three vectors into three channels and apply average pooling with spatial dimension 1x1.

Loss The loss used is the mean square error loss. It is optimized by the Adam optimizer.

Hyperparameters Learning rate was set to 10e-5 (fine-tuning, prevent overfitting). Batch size was set to 32. Embedding dimension is 100.

2.2 Data pre-processing

Embeddings We used the GloVe embeddings. The pre-processing was a simple tokenization with space separations, excluding numbers, leading and ending spaces. All the characters were lowercased. In spite of the loss of some information (for instance, "White House" becomes "white house"), the semantic advantage from the embeddings is still higher compared than if we learn it from scratch ourselves. It is also worth noticing that we

do not freeze our weights, meaning that these semantics can evolve regarding the new context of our train dataset.

The goal is to obtain $\vec{v}_{sentence}$, $\vec{v}_{replaced}$, and $\vec{v}_{replacing}$. The embeddings of the words was straightforward to obtain from the embeddings. In order to obtain $\vec{v}_{sentence}$, we applied a simple masked mean to reject unknown words.

2.3 Results analysis

The number of epochs was determined to avoid overfitting (increasing validation loss): 10 epochs.

The architecture went into many iterations. It was observed that overfitting was very fast. This was handled by setting a very low learning rate and by decreasing the number of fully-connected layers. eLUs activations were motivated by the fact that gradient was vanishing with simple ReLUs. Other final activations were tried, for instance the sigmoid function (with rescaled scores). Sigmoid and hyperbolic tangent functions have high gradients in the [-1, +1] segment, meaning that values will be pushed to the extremes.

We have better results without inserting new words: this might be because we initialize the word embeddings with a normal distribution. Then, the learning process have a fixed learning rate for all the words (very low) meaning that new words might not convergence as fast.

We have better results with sigmoid activation, which can be because there is a stronger demarcation between funny and not funny headlines. In fact, a histogram of the scores show a high amount of not funny headlines (unbalanced grades).

It is also worth noticing that our model rarely predict very funny headlines. It is consistent with the meanGrades distribution of the ground truths. Outputs are over-concentrated in the [0.5, 1.5] area.

3 BiLSTM

3.1 Model

Architecture - For approach 1, we tried to use a BiLSTM model to predict the funniness of an edited new headline. The BiLSTM model used has a hidden dimension of 50 and an embedding dimension of 100. For the embedding of our corpus, we used GloVe embedding of size 100.

Loss - The loss used is the mean square error-loss. It is optimized using the Adam optimizer.

Hyperparameters - The model was trained for 5 epochs using a batch size of 32, using a learning rate of 1e-5.

3.2 Data pre-processing

First, we preprocessed the training dataset which originally contained only the original sentence or the new word. After preprocessing, we can choose to use the original sentence, the edited sentence, the concatenation of original and edited sentences, and finally a new sentence "From *original* to *edit*". For the BiLSTM model we used the latter ("From *original* to *edit*") as input since the best results obtained using this model were with these sentences as inputs. Moreover, the embedding used for the BiLSTM model was the GloVe 100d embedding.

3.3 Results analysis

The model overfitted quite quickly during training, in 3 or 4 epochs only. Same as with the Conv+MLP model, embeddings using GloVe100d ignored around 40% of words wich resulted in poor RMSE (0.55-0.56).

4 BERT Transformer

4.1 Model

Architecture - We also used a BERT Transformer model to predict the funniness of edited news headlines. In particular we used the BertForSequenceClassification, which is a BERT Model transformer with a sequence classification/regression head at the end. Our model was initialized from the pre-trained model 'bert-base-uncased', which was pretrained on English language using a masked language modelling and does not make the difference between upper and lowercase words. In our case, the number of labels is only 1 because it is a regression problem.

Loss - The loss used is the mean square error-loss. It is optimized using the AdamW optimizer.

Hyperparameters - The model was trained using the AdamW optimizer with a learning rate of 2e-5 and an epsilon of 1e-8 for numerical stability. The training of this model took 3 epochs, which is sufficient for our task since the model tends to overfit very quickly.

4.2 Data pre-processing

The input was different depending on the models. First, we preprocessed the training dataset

which originally contained only the original sentence or the new word. After preprocessing, we can choose to use the original sentence, the edited sentence, the concatenation of original and edited sentences, and finally a new sentence 'From "original" to "edit"'. For the BERT transformer model, the best results were obtained using the edited sentence only as input, with the original word replaced by the new one. The embedding used with the BERT transformer model was the 'bert-base-uncased' embedding.

4.3 Results analysis

This model performed significantly better than simpler ones, with experiments showing validation RMSE of 0.53-0.54. In this case also the model tends to overfit very quickly during training, only 2 or 3 epochs are sufficient to train the model. Hyperparameter tuning was done to determine appropriate values for the learning, rate, batch size, epsilon and number of epochs.

5 Approach 2

In this section, no embedded representations or pre-trained models are used.

This approach is based on the assumption that the funny aspect of the data set relies on how absurd it is to find the edited word in the provided context. After trying different combinations of the data (context, edited words, edited sentences, old words etc.) the best results were obtained using as inputs the full original headlines (without tokens) and the edited version of the headlines. After some pre-processing, we compute a TF-IDF representation of the inputs and apply PCA to reduce their dimensionality. Finally, a gradient based regressor, namely XGboost regressor, is tuned and trained to predict the mean grades.

5.1 Pre-processing

We don't work with pre-trained embeddings and, since the data set is small, it is not an easy task to learn efficient ones. Therefore, pre-processing is the key part of this approach.

Even though TF-IDF is not ideal as it produces large vectors, the corpus being rather small allows us to use it. Stop-words helps us reduce the vocabulary and get rid of words that we deem do not play an important role in making the headlines funny. We also use regexp to remove all the numerical values and dates because we consider they have no incidence in the humor of the headlines. Similar words are grouped together thanks to lemmatization, reducing the vector size.

Most of the very funny headlines contain a proper nouns or NNPs in POS tagging. For example, "Trump" which appears on more than one third of the headlines is often part of a humorous headline. Thus, POS tagging could have led to good results if used to give more weight to NNPs or to use them as features. However, some headlines have their first letter capitalized and some have not, making it very difficult to determine NNPs. This method did not work well and unfortunately provided poor results. Instead, it was chosen to use lower case sentences.

Headlines carry a lot of meanings and references in just a few words. They are complex to analyze and would need to consider the position and the context of the words. However, traditional TF-IDF does not take into account the neighbouring words and loses some information. N-grams (for instance 1,2 and 3-grams) could have helped but the Google colab session would crash every time we tried to compute them or apply PCA or an estimator on them.

5.2 Regression

Now that we have a vectorized representation of the data, it is possible to feed it to a regressor in order to predict the mean grades. However, given that a feature vector is very sparse and that there is redundancy, we first apply normalization and PCA operations to reduce the dimensionality of the inputs and make the regressor work faster while keeping 99% of explained variance. The number of features is divided by four down to 3000. We chose XGboost regressor which is based on gradient boosting and is freely available. It works well for large inputs and is very adjustable which is nice to address overfitting.

The parameters (learning rate, tree depth and L1 loss) of the regressor were tuned using 3-fold cross validation. The learning rate and the maximum depth of the tree are paramount to enable afficient learning and prevent overfitting.

5.3 Results

The model achieved to predict reasonably well (RMSE of 0.56 for validation set and 0.57 on the test set) considering that not pre-trained representation or model were used and that the task is not an easy one. However, when testing on individual

headlines, we can see that it still struggles to detect funny swaps.

The regressor was prone to overfit quite quickly and compelled us to use a low learning rate. Some constraints discussed above, such as the size of the data set, or the inability to use N grams or POS tags could have made the predictions better if addressed.

Also, one can notice (see the notebook) that the distribution of the mean grades is unbalanced. Yet, it did not improve the results to artificially augment the number of very funny headlines.

6 Discussion

Below are the results for the different models. As expected, the approach that does not use pretrained models or embeddings gives the worst results and BERT gives the best results. A larger data set or using data augmentation with noise with respect to judges scores could be a way to improve the results. A better fine tuning of hyper parameters and a some more hand processing of the data could also help.

MODELS	RMSE
CNN	
ReLU/w.o. new words	0.5588
ReLU/with new words	0.5600
Sigmoid/w.o. new words	**0.5560**
Sigmoid/with new words	0.5596
BiLSTM	**0.5695**
BERT	**0.5536**
TF-IDF+Boosting	**0.5741**

BERT leads to the lowest RMSE. When looking at the ouput mean grades "by hand", we can observe that there are no very high (> 2) and very low (< 0.5) values. The mass of the distribution is centered near 0.9. This phenomenon is to be related with the initial distribution of the grades in the training set:

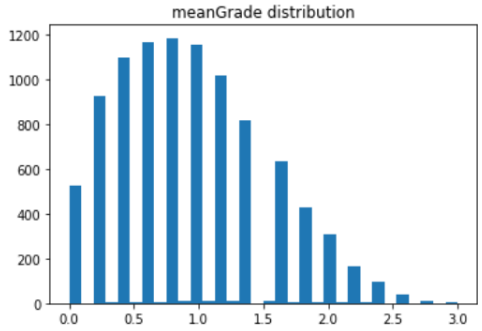

Figure 1: Distribution of the meanGrades on the training set

However, this is not the case for BERT which achieves to predict grades with a higher variance, with for instance a significant number of grades above 2.

A Appendices

XG boost documentation:
https://xgboost.readthedocs.io/en/latest/python/python_api.html

Huggingface for BERT: https://huggingface.co/transformers/model_doc/bert.html