SemEval-2020 Task 7: Assessing Humor in Edited News Headlines - Subtask 1 (Regression)

Salih Tuncer
Deep Learning for Natural Language Processing – SS21 – Philipp Cimiano and Philipp Heinisch

August 31, 2021

1 Introduction

In this paper we deal with the understanding of humor on the part of artificial intelligence. While we as humans already have difficulties in understanding, let alone perceiving, humor, we want to address the challenge of training models that, in the best case, have an understanding of humor. The complexity of humor lies in the fact that it is highly dependent on certain conditions. These conditions can be, for example, general understanding, mood or the ability to create connections. (Hossain et al., 2020)

Nowadays, state-of-the-art models are quite capable of judging whether sentences are funny or not. These can in turn be used for chatbots or virtual assistants. (Annamoradnejad, 2020) However, we would like to go a step further and teach our models to rate sentences on a scale according to their funniness.

We use pre-trained models and datasets that are publicly available on the Internet and adapt our model to our problem by means of fine-tuning. The dataset called "Humicroedit" consists of approximately 15,000 news titles that have been shared on social media such as Reddit (reddit.com).

One word from the title is replaced by another and evaluated by the jury members. Each sentence is assigned to five jury members. You can choose between the numbers zero to three. The number zero stands for "Not Funny At All", one for "Only Slightly Funny", two for "Funny" and three for "Very Funny". See Table 1. The resulting ratings as well as the mean value are documented individually for each headline. (Hossain et al., 2020) The model adapted to the problem then has the task of receiving a sentence and correctly assigning it to one of the four categories.

In this work, we first deal in detail with a scientific work that has already dealt with this topic. This gives us an overview of what the current state of knowledge about this problem is, as well as the approach to it. Afterwards we justify our choice of methodology, our model architecture and a small introduction to our model choice and its advantages and disadvantages. Later we compare our test results with those of the scientific work in detail and justify the results. Finally we draw a conclusion and present what we gain from it.

Number	Funniness
0	**Not Funny At All**
1	**Only Slightly Funny**
2	**Funny**
3	**Very Funny**

Table 1: Funniness categories

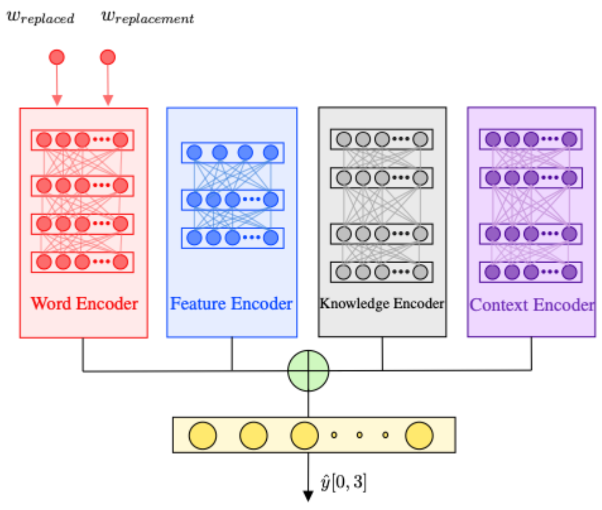

Figure 1: model-architecture from (Jensen et al., 2020)

2 Related work

Just like us, Jensen and the others have dealt with the same subject as we have, but their approach is very different from ours. They used two different architectures. The first architecture contains three different encoders, namely a Word /, Feature / and a Knowledge encoder. The second architecture, which can be seen in figure 1, also includes a Context encoder based on the Albert model (Lan et al., 2019).

The Word Encoder consists of three layers and accepts both the edited word and the replaced word. The task of the encoder is now to encode the individual words in their appropriate form and concatenate them at the end.

The Feature Encoder, on the other hand, is much more complex and has four different features. The first feature looks at the position of the replaced word, as it becomes clear in the dataset that the further to the end the replaced word is, the funnier the headline. (Hossain et al., 2020) A number between 0 and 1 is returned for this. The second feature is based on a similar knowledge. Here you can see how long the headline is. The longer the headline, the funnier it is according to (Hossain et al., 2020) Here, too, a number between 0 and 1 is returned accordingly. The third feature deals with the phonemes of a word. The annotators of the headlines tended to use similiar-sounding words to replace the old word. (Hossain et al., 2020) Here the Levenshtein distance between the two words is calculated. The last feature calculates the cosine distance of the two words. According to (Hossain et al., 2020), sentences are often funny in which the relative distance between the original word and the replaced word is large.

The Knowledge Encoder looks for already known entities that are available in the NELL database and either returns the embedding, if it is available in the database, or a null vector. The second architecture also has a Context Encoder based on the Albert Model, which is used to reduce the size of the representation. In the end, both architectures only have a linear regression layer as an output.

Team	RMSProp	Adam
(Jensen et al., 2020)	-	**0.5434**
Our model	**0.4569**	0.4944

Table 2: Jensen and the others in comparison to our scores on the test set

3 Method

In the dataset for each edited headline, we have a floating point number as the mean rating in the interval [0,3]. Our task is now to use our model to predict reasonable floating point numbers for headlines from the test dataset. It is therefore a regression problem. When we visualize the problem, we have different points in space in a coordinate system. The task of the model is to draw a straight line that has the lowest possible error rate.

We have chosen the architecture of the model accordingly. Instead of building a model from scratch, we fall back on pre-trained models. We decided on the 'TFAlbertForSequenceClassification' model from Tensorflow, which specializes in the classification of sequences and is therefore designed precisely for our problem. All we have to do is fine-tune this to the problem using our dataset.

Our model consists of an AlbertMainLayer, a Dropout Layer and a Dense Layer. The Dropout layer prevents overfitting while the Dense layer is used for prediction. The AlbertMainLayer in turn consists of further custom layers and so on which we will not go into further.

Our Albert model, which is a lite version of the Bert model (Devlin et al., 2018), has approximately 11 million parameters, 80 percent fewer parameters than the Bert model. (Lan et al., 2019) Accordingly, our model is many times smaller on the hard drive. The number of parameters already shows that this is a particularly complex model. Therefore it is already possible to achieve considerable values with just fine-tuning.

A big disadvantage is that if we want to make changes to the architecture that should bring positive results, we have to make a lot more changes in contrast to small and simple architectures.

4 Evaluation

As an evaluation metric, we use the Root Mean Square Error, which tells us how far the data points are from our regression line. This is used by default for regression tasks and is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(\frac{d_i - f_i}{\sigma_i}\Big)^2}(1).$$

We used two different optimizers, the Adam Optimizer and the RMSProp. As you can see in the table 2, we get an RMSE of 0.4569 while with Adam we get an RMSE of 0.4944. We did not expect that RMSProp has actually a lower RMSE than Adam because Adam is a combination of Momentum and RMSProp and should therefore result in lower error rates. However the difference of four percent is considerable.

In comparison Jensen and the others used only the Adam Optimizer. They achieve an overall best value RMSE of 0.5434 and are thus 0.0865 above our best model. So they have an approximately 19 percent higher error rate than we have.

Jensen and the others went to great lengths with their well-thought-out architecture by defining various encoders and linking them to one another. The results also show that they harmonize well with each other.

Our approach shows that it is also possible, with a simple architecture like we have, to achieve good results as well.

5 Conclusion

In this paper we tried to train a model who is able to evaluate the funniness of headlines. We implemented this using the Albert Model, which specializes in classifying sequences. We used the RMSE as a metric and Adam and RMSProp as optimizers. With the Optimizer Adam we achieved an RMSE of 0.4944 while with RMSProp we even reached 0.4569, which are quite good values compared to the other groups.

So we see that with little effort it is possible to build a model, which is able to evaluate the funniness of individual headlines. If we look at the NLP task in general, we see that there is still a lot of room for improvement. For example, you could add additional custom layers that are specially adapted to our problem. You could also take different pre-trained models and combine them to build a larger construct.

References

Annamoradnejad, Zoghi (2020). *ColBERT: Using BERT Sentence Embedding for Humor Detection.* `https://arxiv.org/abs/2004.12765`.

Devlin, Chang et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* `https://arxiv.org/abs/1810.04805`.

Hossain, Krumm et al. (2020). *Stimulating Creativity with FunLines: A Case Study of Humor Generation in Headlines.* `https://arxiv.org/abs/2002.02031`.

Jensen, Kristian Nørgaard et al. (2020). *Buhscitu at SemEval-2020 Task 7: Assessing Humour in Edited News Headlines Using Hand-Crafted Features and Online Knowledge Bases.* URL: `https://aclanthology.org/2020.semeval-1.104`.

Lan, Chen et al. (2019). *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.* `https://arxiv.org/abs/1909.11942`.