Assessing Humor in Edited News Headlines

Chalmers University of Technology
Data Science and AI

Anjali Poornima Karri
anjalip@student.chalmers.se

Prashant Kumar
kumarpr@student.chalmers.se

Kuvalaya Datta Jukanti
kuvalaya@student.chalmers.se

Swaathy Sambath
swaathy@student.chalmers.se

Abstract

The recent progress in machine learning and natural language processing techniques have led people to find new ways to utilize them. With an idea inspired from a competition task SemEval-2020 "Assessing Humor in Edited News Headlines", the motivation for this project is to build a model that has the ability to predict the funniness of edited headlines, thereby providing a way to automatically detect humor. This paper describes the implementation of three different deep learning models namely, Feed Forward Neural network, Bidirectional Encoder Representations from Transformers (BERT) and Recurrent Neural Network on the humor dataset and compare the results. The results show that the BERT model outperforms the other models.

Keywords - Humor detection; Deep Learning; SemEval-2020; Humicroedit;

1 Introduction

"Humor is the great thing, the saving thing after all. The minute it crops up, all our hardness-es yield, all our irritations, and resentments flit away, and a sunny spirit takes their place", Mark Twain said [1]. Humor is an important constituent of human communication, and every automatic system aiming to reproduce human intelligence will have to eventually work on developing capabilities of detecting or generating humorous content. [7]

In the field of artificial intelligence, humor detection still continues to be a challenging task. However there has been a steady and slow progress in the research. While there has been some advances in automatic humor detection (e.g., [8] [5] [4]), computerized humor generation ([10], [9]) has seen less progress [6], given that humor involves a strong domain knowledge and a complex understanding of the relation between words. Even in the real world, humans sometimes face difficulties in being funny or detecting humor. Humor happens at various intensities where certain jokes are much funnier than others, and therefore developing a system that has the ability to assess the funniness in a given text is more complicated as the system should be able to perceive the relationship across entities and objects. A system like that can used for various applications such as in a generate-and-test scheme to generate humorous texts and rank them by funniness, automatic fill in the blanks in etc [7].

2 Methodology

The proposed system describes various deep learning algorithms for determining the funniness of the edited headlines. the implementation is structured into four modules namely, Analysing the data, Preprocessing, Model Training and Evaluation. The preprocessing stage involves techniques to clean and format the data in such a way that the model can use it, while the model training phase involves the use of different deep learning models to train and test the data. The evaluation module measures the error of the model in predicting the quantitative data.

2.1 Data

The text corpus carried out for this work is "Humicroedit" [2], a novel dataset for research in computational humor. The dataset contains news headlines collected from news media posted on Reddit. Short edits are applied on the text to turn it from non-funny to funny. These edits and annotations were done by qualified experts from Amazon Mechanical Turk. Figure 1 shows a sample example of how these headlines were edited.

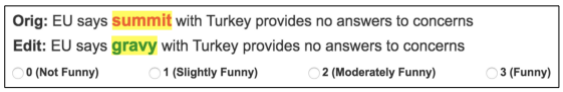

Figure 1: Headline editing task

Each edited headline is scored by five judges where they assign a grade numerically on a scale of 0-3 of funniness scale, where 0 being non-funny and 3 being the most funny. The quality and the ground truth funniness is determined by computing the mean of the five funniness grades. The resulting dataset contains 15,095 edited headlines with numerically evaluated humor. Figure 2 shows an sample of a few edited headlines graded based on the funniness scale.

Original Headline	Substitute	Grade
Kushner to visit Mexico following latest Trump tirades	therapist	2.8
Hilllary Clinton Staffers Considered Campaign Slogan `Because It's Her Turn'	fault	2.8
The Latest: BBC cuts ties with Myanmar TV station	pies	1.8
Oklahoma isn't working. Can anyone fix this failing American state?	okay	0.0
4 soldiers killed in Nagorno-Karabakh fighting: Officials	rabbits	0.0

Figure 2: Edited headlines with their funniness rating

This method of assessing humor enables the development of systems that can automatically estimate the degree of humor in text. In addition, these existence of these factors will help in developing tools that are capable of deeper understanding and reasoning. [6]

2.2 Preprocessing

Preprocessing the input data is the first and an important step for building a good NLP application. The purpose of preprocessing is to put the data into a predictable and analyzable form. The dataset contains of five columns namely the ID, the original sentence, single word edits, grades and the mean of the grades. The word that has to be replaced in the sentence is represented by (< and />). Using the regex library in python, the original word and the edited headlines are obtained.

2.2.1 Tokenization

Tokenization is the first step in any NLP pipeline. A tokenizer breaks the raw text into chunks of information which can be considered as discrete elements. These elements can either be in the form of words or sentences. There are various libraries to perform tokenization such as NLTK, Gensim, Keras etc. In our case, the word tokenization was performed to tokenize the original headlines and the corresponding edited headlines using the NLTK library. This helped in understanding the context and interpret the meaning of the text by examining the sequence of words.

2.2.2 Building Vocabulary

After extracting sequence of terms from the data, the next preprocessing task is to build a vocabulary for our Language Model. This is done by using "word2idx". It is a dictionary that has all the unique words as keys with a corresponding unique ID as values. This helps in indexing the words for tokenized original headlines and the new words. An extra pad token is also included. By using this technique, the vocabulary for the textual data was created.

2.2.3 Word Embedding

Word embedding is one of the most important techniques for preprocessing the text in NLP applications. It is a technique where words or phrases from a vocabulary are mapped to vectors of real numbers. In our case, a function is implemented which uses the vocabulary to create a numerical representation of the elements. These numbers represent the tensors that is used in the deep learning model. This is done by using the PyTorch machine learning library. The data loader for the models is constructed with the vectorized original headlines, edited headlines and the labels (the mean score of funniness) as shown in Figure 3.

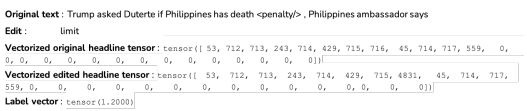

Figure 3: Sample example of the preprocessed structure

2.3 Modeling

Deep Learning models are said to provide promising results on natural language textual data. The main advantage of deep learning models over other tradition machine learning algorithms is how it can deal with the features. In addition, the representation of features is also not computationally efficient as traditional machine learning algorithms use sparse matrix to represent features

which leads to a very high dimension of feature matrix. On the contrary, the deep learning approaches represent the features in distributed manner, for example, using word embedding, Word2Vec etc. Our dataset is sequential in nature which requires contextual learning to solve the problem. Therefore, the idea was to use three deep learning models namely, Recurrent Neural Network (RNN), Feed Forward Neural Network (FFNN) and Bi-directional Encoder Representation for Transformers (BERT).

2.3.1 Feed Forward Fully Connected Neural Network (FFNN)

Feed forward neural networks, also known as multi-layer perceptrons is a network with multiple layers connected to each other. The connections between the nodes does not form a cycle. The main idea behind using FFNN is to learn the intermediate representations layer by layer to pass to inputs in the form of original headlines and edited headlines.

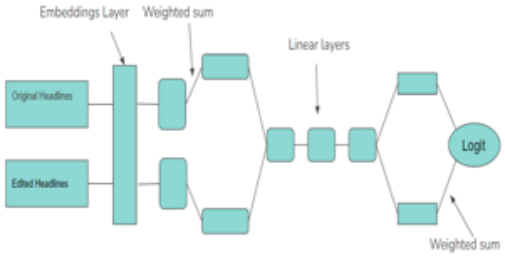

Figure 4: FFNN architecture

The architecture starts with two inputs to the embedding layer where each word is transformed into a fixed dimension embedding in the headlines. The model then computes the weighted sum of all the headlines to make a representation vector for each headline. The model is then equipped with three linear layers. A ReLU activation function is applied on the output of first two hidden layers. Later, a weighted sum is calculated over top of it which returns a vector in the dimension *(original headlines number,1)* and then compute a humor value. Figure 4 gives a graphical view of the above described architecture. The FFNN architecture built in our case uses an Adam optimizer with a learning rate of 0.145. The learning rate is scheduled using a CosineAnnealingLR scheduler with warm-up steps of 30. In total, the model was trained for 50 epochs.

2.3.2 Recurrent Neural Network (RNN)

Recurrent neural networks (RNN) is a state-of-the-art algorithm essentially used for sequential data. It has the ability to remember the input in its internal memory which makes it a robust and powerful algorithm. In addition, the network can form a deeper understanding of the sequence and hence can learn the context.

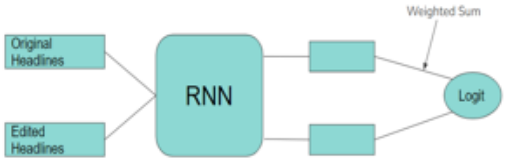

Figure 5: RNN architecture

Figure 5 describes the RNN architecture built for our data. It consists of two inputs fed to a single layer of bidirectional RNN unit as we are performing a regression task. The pre-trained embedding is obtained by training a classification task with the target as rounded scores. These pre-trained embedding are used while training a Bidirectional RNN unit. A linear layer takes the final hidden state and feeds it into a fully connected network. The inputs to RNN unit is in the form of tensor and then a row wise weighted sum is computed over the output of the RNN unit which provides a vector in the the dimension of *(original headlines number,1)*. Dropouts are used to prevent over fitting. The RNN architecture built in our case uses an AdamW optimizer with a learning rate of 1e-4. In total, the model was trained for 50 epochs.

2.3.3 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is based on the concept of bidirectional training of transformer, is a popular attention model. It is widely used language model for NLP tasks. It consists of two parts, encoder and decoder which depends on the type of problem. The model is capable of learning contexts by the use of surroundings. Another advantage is that it provides a very fast hyper parameter tuning, depending on the problem.

Unlike the tokenizer used for the other models, a BERT tokenizer is used in this case to break down the sentences into useful chunks. Next, the original news headlines are concatenated with new words to create a input for the model. Figure 6

shows an example of how a sentence is encoded. The data loader consists of the encoded input tokens, attention mask for the BERT model, token type ids and the labels. The pre-trained BERT model from HuggingFace is used and is fine tuned with our data for 2 epochs. AdamW optimizer is used for this model with a learning rate of 8e-3. Weight decay is applied to all parameters other than bias and layer normalization terms. The output of the cls label is used as the score for humor value.

Encoded input to the model : '[CLS] pentagon claims 2, 000 % increase in russian trolls after bowling strikes. what does that mean? [SEP] bowling [SEP] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]'

Figure 6: Encoded Input

3 Evaluation, Results and Discussion

3.1 Evaluation Metric

The metric used for determining the quality and evaluation of the system is by measuring the root mean squared error (RMSE). Root mean square error is a commonly used measure for evaluating the quality of predictions. It measures the error of a model in predicting quantitative data. The RMSE score is computed by using the formula,

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values
$y_1, y_2, \ldots, y_n$ are observed values
n is the number of observations

Figure 7: Root Mean Squared Error

3.2 Results and Discussion

The results of all the models implemented on the data is shown in Figure 8. The RMSE score obtained acts as the final metric in determining which model has performed the best. To summarize the results, the fine-tuned **BERT** model has proved its efficiency over the other models as it has gained the RMSE score of **0.515**. On the contrary, the error scores of Feed Forward Neural Network and the Recurrent Neural Network are much higher. The error score of RNN is marginally better than the FFNN but the time complexity of RNN is significantly higher than the FFNN. These scores obtained are reasonable when compared to the results on the competition page [3].

Model	Test Loss	RMSE score	Epoch Time(Avg)
Feed Forward Neural Network	0.580218	0.579972	1.768 sec
Recurrent Neural Network	0.573476	0.575363	5.085 sec
BERT model	0.514357	**0.515648**	156.33 sec

Figure 8: Results of the models implemented

4 Conclusion

Analyzing the funniness in news headlines or in any text and computing the degree of the funniness is a challenging and a critical task. This work is focused on applying different deep learning models and measure the quality of the predictions. The results showed that the fine-tuned BERT model is the better model for our data and study.

References

[1] `http://www.twainquotes.com/Humor.html`.

[2] `https://cs.rochester.edu/u/nhossain/humicroedit.html`.

[3] `https://competitions.codalab.org/competitions/20970#results`.

[4] Dario Bertero and Pascale Fung. A long short-term memory framework for predicting humor in dialogues. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 130–135, San Diego, California, June 2016. Association for Computational Linguistics.

[5] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, page 107–116, USA, 2010. Association for Computational Linguistics.

[6] Nabil Hossain, John Krumm, and Michael Gamon. "president vows to cut <taxes> hair": Dataset and analysis of creative text editing for humorous headlines. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 133–142, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. SemEval-2020 task 7: Assessing humor in edited news headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758, Barcelona (online), December 2020. International Committee for Computational Linguistics.

[8] Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A large self-annotated corpus for sarcasm, 2018.

[9] Saša Petrović and David Matthews. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[10] Oliviero Stock and Carlo Strapparava. Hahacronym: Humorous agents for humorous acronyms. *Humor-international Journal of Humor Research - HUMOR*, 16:297–314, 01 2003.