YNU-HPCC at SemEval-2020 Task 7: Using an ensemble BiGRU Model to evaluate the Humor of Edited News Titles

Joseph J. Tomasulo, Jin Wang and Xuejie Zhang
School of Information Science and Engineering
Yunnan University
Kunming, China
Contact:{jtomasulo, wangjin, xjzhang}@ynu.edu.cn

Abstract

This paper describes a ensemble model designed for Semeval-2020 Task 7. The task is based on the Humicroedit dataset that is comprised of news titles and one-word substitutions designed to make them humorous. We use BERT, FastText, Elmo, and Word2Vec to encode these titles then pass them to a Bidirectional GRU with attention. Finally, we used XGBoost on the concatenation of the results of the different models to make predictions.

1 Introduction

The Humicroedit dataset was created for research in computational humor (Hossain et al., 2019). The authors employed Amazon Mechanical Turk annotators to edit a single word of some fifteen thousand news headlines in order to make them funny. Five other annotators then graded each edited title on a scale of 0-3, with 3 being the funniest. Titles, despite being very short (the longest were around twenty words), convey a lot of information. Using one word edits is a way to make minimal changes for an expressed purpose. These two factors make the Humicroedit dataset a useful tool for studying computational humor.

During the competition, the organizers published the Funlines dataset, which improved on Humicroedit in terms of cost per title by gamifying the process: they provided cash rewards for the best annotators (Hossain et al., 2020b). The performance of annotators was measured on a mix of how funny their edits were and how well their grades tracked the overall average. Players were able to improve in both categories and the authors posit that this was due to the availability of live feedback. On average, the Funlines dataset is funnier than Humicroedit and there was better agreement on the grades. The Funlines dataset was made available to participants of the competition in January.

There were two sub-tasks for SemEval Task 7: 1) for each headline and its edit, predict the mean of the grades assigned by the five annotators; 2) decide which of two title–edit pairs is funnier. We use a few popular pretrained language models in a BiGRU ensemble for the first task, and we use those results for the second task.

The rest of the paper is organized as follows. Section 2 presents a brief review on humor assessment and some NLP tools. Section 3 presents the BiGRU ensemble model in detail. Section 4 contains some discussion of the results and section 5 is a short conlcusion.

2 Background

Humor presents some challenges that other NLP tasks like entailment and analogy, for example, do not: it is highly contextual and subjective. Because of the latter, we may prefer to ask how funny something is or to whom is it funny, and when, over a question like “Is this funny? Yes or No”. So, attempts at computational humor lend naturally to regression tasks.

Continuous representations have proven to be a good basis from which to transform a natural language task into a computational one. In order to produce continuous representations, many researchers have relied on the distributional hypothesis, which states that the meaning of a word is the assortment of

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 871–875
Barcelona, Spain (Online), December 12, 2020.

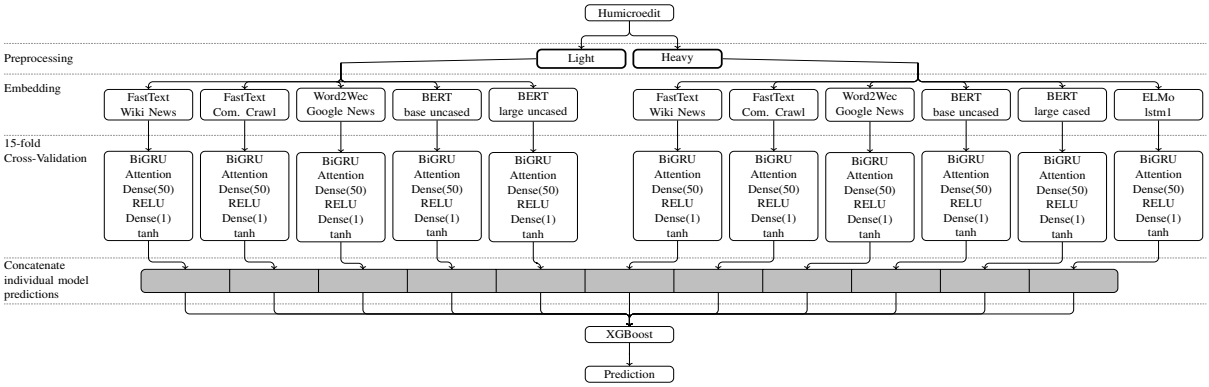

Figure 1: The structure of the model. Light preprocessing removes only irrelevant context while heavy preprocessing removes the context as well as all punctuation except for apostrophes.

contexts in which it appears. Here we briefly mention the four language models for producing continuous representations that we use for Task 7.

Word2Vec trains a shallow neural network to predict a word from its context (skip-gram) model or the context from a word (continuous bag-of-words) (Mikolov et al., 2013a; Mikolov et al., 2013b). FastText does something similar but trains on characters, whose n-grams are averaged to get word vectors (Bojanowski et al., 2017). This allows fastText to perform better than word2vec on out-of-vocabulary (OOV) words.

ELMo trains continuous representations by using an LSTM to predict the next token in a sequence of tokens, and it does this in reverse too, with a bidirectional LSTM (Peters et al., 2018). The representations are then combined linearly, creating context dependent word representations.

BERT uses an attention style, multi-level encoder-decoder structure called a transformer, which can model relationships between tokens that are farther apart than can LSTMs (Devlin et al., 2019). The BERT transformer uses masking on about 15% of tokens—sub-word entities of a non-fixed length. In pretraining, BERT also asks the model to predict whether two sentences are sequential. BERT allows users to select vectors from specific levels of the transformers and combine them in various ways. Another feature, and one that is relevant here, is that the user can choose to not pool the vectors at all, and this results in word vectors.

3 System Overview

As shown in Figure 1, this system is a stacked Bidirectional GRU ensemble. The first level uses cross validation on 11 different embeddings. Predictions of the first level are passed to an XGBoost regressor to get the final output. The results for sub-task 1 were directly used to decide which of the two headlines was funnier for sub-task 2, so the following system is designed simply to predict how funny the annotators thought the edited headlines were.

3.1 Data and Encoding

We regret that we were unable to improve performance using the Funlines data. In retrospect, the humor ratings have different distributions in the two datasets and it is possible that some kind of calibration was in order (Hossain et al., 2020a). We also left out the replaced word from the original title, and instead just encoded the edited titles with four off-the-shelf pretrained models.

Embedding-as-Service[1] was used to get FastText and Word2Vec vectors for the titles. FastText vectors came from models trained on Wiki News and Common Crawl, both with dimension 300. Word2Vec was trained on Google news, again with 300-dimensional vectors. These were very straightforward and performed consistently on our task.

[1]https://github.com/amansrivastava17/embedding-as-service/tree/master

For BERT, Huggingface and HanXiao's Bert-as-Service were very helpful for quick encoding.[2][3] Additionally, we experimented with a Kaggle dataset called "All the News" whose articles were published shortly before the articles corresponding to the news headlines in Humicroedit.[4] We expected that starting from a BERT checkpoint and continuing pretraining on this news data would improve the performance but initial tests were not promising, so we dropped the idea. The best performance was achieved using no pooling, which gives something like word vectors. The new models with Whole-Word-Masking didn't do as well as the base and the large models, so we left them out. On our holdout set, it was unclear whether cased was better than uncased, so we used both. Finally, we also tried fine-tuning various BERT models on Humicroedit, but the results weren't as good as the BiGRU from Keras.

The last language model we used was ELMo, for which we took advantage of the relevant Tensorflow Hub module. The LSTM1 output was consistently better than the rest, so that was used exclusively.

Power Transformation and the Standard Scaler from Scikit-Learn were used to normalize the data. While the latter was a bit better for individual models, they performed equally in the ensemble. For submission we used the Box Cox method of the Power Transformation.

Parameter Name	Value
patience	2
max epochs	60
cross val. folds	15
hidden dimensions	120
recurrent dropout	0.25

Table 1: BiGRU

Parameter Name	Value
objective	squarederror
gamma	0
min_child_weight	5
colsample_bytree	0.2
learning_rate	0.0055
max_depth	4
reg_alpa	0.2
subsample	0.1
n_estimators	300

Table 2: XGBoost

3.2 Bidirectional GRU

Broadly, the first level consists of a Bidirectional GRU, some form of attention, a dense layer with RELU actiavtion and 50-dimensional output, and another dense layer with 1-dimensional output and hyperbolic tangent activation. Table 1 shows some of the relevant parameters.

Two different forms of attention were used. For BERT and Elmo, attention vectors were calculated from the matrix output of the BiGRU. For FastText and Word2Vec, it worked better to use Global Average Pooling on the output of the BiGRU and use that as a context vector for attention on the matrix output of the BiGRU.

Other than this, the loss functions were also different: logcosh for FastText and Word2Vec versus mean squared error for BERT and Elmo. Both setups used the stock adam optimizer provided by Keras.

We used Keras to try an LSTM, a CNN, CNN-LSTM, multi-channel CNN, GRU, and a Capsule Net. We found that BiGRU's had the highest individual model performance and adding any other model reduced ensemble performance.

The hyperbolic tangent activation used on the output layer returns values between -1 and 1, so when we applied the reverse power transform, we were not using the full range (which was about -2.5 to 2.5 after scaling with the power transformation), but we found that re-scaling the output reduced the performance.

3.3 Ensemble Learning

We ran cross-validation on each embedding six times and saved the results from the model that performed the best. Training models on all eleven embeddings using 15-fold cross validation, and doing each six

[2]https://huggingface.co/

[3]https://github.com/hanxiao/bert-as-service

[4]https://www.kaggle.com/snapcrack/all-the-news

Embedding	Preproc	B1	B2	B3	B4	RMSE	Accuracy
FastText (Wiki News)	heavy	**0.8512**	**0.7280**	**0.6435**	**0.5838**	**0.5378**	0.6275
FastText (Com. Crawl)	heavy	0.8768	0.7506	0.6593	0.5945	0.5442	0.6153
FastText (Wiki News)	light	0.8750	0.7443	0.6529	05896	0.5391	0.6214
FastText (Com. Crawl)	light	0.8612	0.7345	0.6477	0.5861	0.5388	**0.6396**
W2V (Google News)	heavy	0.8754	0.7424	0.6528	0.5884	0.5395	0.6271
W2V (Google News)	light	0.8772	0.7481	0.6603	0.5968	0.5466	0.6233
BERT (base uncased)	heavy	0.8820	0.7527	0.663	0.5951	0.5429	0.6164
BERT (large cased)	heavy	0.9011	0.7678	0.6764	0.6082	0.5569	0.6088
BERT (base uncased)	light	0.8741	0.7472	0.6571	0.5905	0.5395	0.6313
BERT (large uncased)	light	0.8871	7555	0.6623	0.5935	0.5410	0.6107
ELMO (lstm1)	light	0.8772	0.7500	0.6605	0.5933	0.5418	0.6260
Simple Average	-	0.8644	0.7332	0.6417	0.5746	0.5224	0.6377
XGBoost Ensemble	-	0.8118	0.7017	0.6206	0.5626	0.5181	0.6469
Baseline	-	0.9860	0.8309	0.7225	0.6402	0.5747	0.4905

Table 3: Top section has the individual model results including both bucket and overall performance. Heavy preprocessing removes punctuation; light does not. The bottom section has ensemble methods and the baseline. This is a different run than was submitted, so the scores are slightly different, but the models presented here are the same as used in the submission that placed third. The simple average just averages the predictions of the models. The baseline predicts the mean of the train set everytime.

times, resulted in a total runtime of about four hours on a Nvidia 2060 SUPER.

For each model, we concatenated the predictions on the folds left out of training. Then we concatenated these predictions from each model, and fed this to a few XGBoost regressors. We tried using AdaBoost, Random Forest, and SVR but XGBoost was consistently the best. Still, sometimes different hyperparameters performed better so we kept four of the best performing hyperparameter settings and selected the one with the highest score on the dev set at run time to make a prediction on the test set. Table 2 shows the parameters for the model that made the best predictions on the ensemble used in the competition.

3.4 Experimental Setup

For preprossessing, we remove things like " – The New York Times" or " – live updates" that somtimes appeared in the titles. This was all the preprocessing we did for about half of the models and it resulted in a title length of 30 tokens. For the other half, we removed all puncuation, resulting in 21-token titles. We experimented with this only a little, and perhaps left something on the table here.

We sorted the data in the train set so that each fold in cross validation had a distribution of y values that was representative of the whole train set. This seemed to make the scores more consistent over different runs, but did not noticeably improve them.

It may be worth noting that due to the competition rule that only the last submission would be counted, we never actually use all the available data for training. We didn't want to rely on a model whose performance was not verified on a holdout of the train set.

The organizers graded the first sub-task using root mean squared error and that was what we used throughout testing. A bucket performance metric was somewhat useful during development, it independently scores then bottom 10% (least funny) and top 10% (most funny) as the first bucket, the bottom 20% and the top 20% as the second bucket, etc.

4 Results and Discussion

Our model placed third in both sub-tasks with an RMSE of 0.51737 for the first and an accuracy of 0.65906 for the second. Over the course of this competition, much of our improvement was due to time consuming trial-and-error in composing the ensemble.

It is surprising to us that FastText and Word2Vec have individual model scores on par with the model using BERT encodings. In fact, as Table 1 shows, the best performing individual model comes from using FastText and the best model using Word2Vec performed better than the best of BERT, although it was only by as small margin.

The simple average of the results produced by each model was much better than any individual model, but not as good as XGBoost. The baseline of predicting the mean is actually in the ballpark of some individual models we tried.

To evaluate the utility of this project, it might help to compare any model with human performance, perhaps using the Funlines data, but we reserve this idea for later. Another idea to pursue is to make this into a classification task: humans, especially those who read news, would presumably be pretty good at deciding whether a title has been edited for the purpose of making it funny.

5 Conclusion

This project used a BiGRU ensemble with XGBoost to predict how funny Amazon Mechanical Turk annotators found edited news headlines. A third place finish on both subtasks of the related SemEval Task 7 was achieved largely by using a large number of models for encoding the titles, and by tuning an ensemble for this task.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61702443, 61966038 and 61762091. We would like to thank PengBo of the HPCC research laboratory at Yunnan University, who provided useful guidance in the early stages of this project. In fact, many lab members provided ideas that we tested out, and the ensemble structure was due to this collaboration. We are grateful to Peter Tomasulo and Yunnan University for providing hardware.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Nabil Hossain, John Krumm, and Michael Gamon. 2019. "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. Association for Computational Linguistics.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020a. Semeval-2020 Task 7: Assessing humor in edited news headlines. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain.

Nabil Hossain, John Krumm, Tanvir Sajed, and Henry Kautz. 2020b. Stimulating creativity with FunLines: A case study of humor generation in headlines. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 256–262, Online, July. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. cite arxiv:1802.05365Comment: NAACL 2018. Originally posted to openreview 27 Oct 2017. v2 updated for NAACL camera ready.