Towards an Automated System for Rating Humorous Content in Textual Documents

Chirag Daryani
cdaryani@ualberta.ca

Varshini Prakash
vprakash@ualberta.ca

Abstract

Automatic humor assessment continues to remain a challenging task. This paper describes our system that can rate humorous content in textual documents, designed to solve Task 7 (Subtask 1) of SemEval 2020 on 'Humor, Emphasis, and Sentiment. We experiment with various word embedding techniques in combination with different computational models: Linear Regression, Support Vector Regressor, Random Forest Regressor, Ensemble Models.

1 Introduction

Humorous experiences provide comic relief, amusement and provoke laughter. Humor is an integral part of life for people of all ages and cultures. That being said, humor is often subjective and is ultimately decided by one's personal preferences. Humor can also depend on a variety of factors such as location, culture, maturity, level of intelligence, knowledge, and so on. This makes it a challenging task for machines to comprehend humor. Additionally, there are more sophisticated forms of humor such as satire and sarcasm that require an understanding of factual knowledge, contextual and social meanings. The artificial intelligence academia has made slow but steady progress in the assessment of humor. Some of the earliest works on computational humor include the work done by Hempelmann et al. (2006). Their system generated humor based on ontological semantics. Mihalcea and Strapparava (2005) demonstrated through automatic classification techniques that humorous and non-humorous texts could be distinguished from each other. In recent times, Hossain et al. (2019) curated a dataset of edited headlines with a score assigned for each headline representing its funniness. This dataset was later used for SemEval 2020 as a shared task (Hossain et al., 2020). This data has provided researchers with even more motivation to develop highly successful computational models to assess humor.

Towards this goal of developing intelligent systems that can better recognize humor content from text, the focus of our current work is on developing the solution for the SemEval-2020 Task 7 (Assessing Humor in Edited News Headlines) Subtask 1. In this task, we try to predict the mean humor rating for news articles that were edited by replacing a word in their original content. Our aim would be to develop a sentence-pair regression model that takes both the original news headline and the edited news headline as input and then outputs a prediction of a mean funniness rating in the range of 0 to 3 for each edited news article. The predicted real-valued scores would represent the humor content as follows: 0 represents "Not Funny", 1 represents "Slightly Funny", 2 represents "Moderately Funny" and 3 represents "Very Funny".

By measuring the amount of humor content, we can then understand what exactly makes a particular text piece funny, give ranking to multiple pieces of edited texts based on their funniness rating, and then can also design recommendation systems for the same. We can even use the system in humor generation applications (Hossain et al., 2019). Therefore, solving this task of humor content evaluation from text articles can help us gain much more understanding of the language and this is why we selected this task for our project.

2 Related Work

Previous work towards humor recognition typically used specific aspects of linguistic features in combination with traditional text classification models such as Naive Bayes (Mihalcea and Strapparava, 2005). Some of the earliest works have also tried to leverage patterns of a type of jokes, ones that include wordplay as a component (Tay-

lor and Mazlack, 2004). With the emergence of Deep Learning, De Oliveira and Rodrigo (2015) used recurrent neural networks (RNN) and convolutional neural networks (CNN) to more accurately model the sequential nature of Yelp reviews to detect humor. While Chen and Soo (2018) used CNNs and Highway Networks to classify humor by maintaining a focus on puns and short jokes. Another work by Yang et al. (2015) extracted the semantic representation of the sentences by using the averaged word embeddings from Word2Vec as features. They achieved the best evaluation scores by using a combination of Word2Vec and HCF (Humor Theory driven Features and K Nearest Neighbours features). Hossain et al. (2020) suggests that the dominant teams participating in this task have previously used pre-trained language models along with context-independent word embeddings. The combination of hyperparameter-tuned variations of these models through an ensemble learner using regression has also proved to be successful in the past.

We, therefore, decided to take a similar path for the implementation of this project. We chose to experiment with various word embedding techniques for generating numeric features for our text. We also decided to utilize machine learning models over pre-trained deep neural network models because they are more interpretable and can provide reasonable explanations behind the prediction results which is currently lacking in the related work mentioned above.

3 Methods

The main task we are solving in this project is a **sentence-pair regression** problem in which given the original news headline and the replacement word, we want to predict the humor level in the edited news headline. As mentioned above, the core method we employ for solving this task is experimenting with various word embedding techniques in combination with different types of machine learning models trained on the input dataset. To do this we perform three main steps which are data preprocessing and preparation, model creation and tuning, and finally model evaluation. We can visualize the high-level view of the system in the figure below.

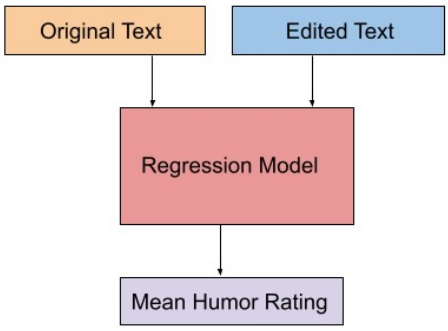

Figure 1: Basic System Architecture with Input and Output

3.1 Dataset

We use the dataset called "Humicroedit" (Hossain et al., 2019) which was provided by the organizers of SemEval. It was created by taking news headlines and making micro-edits i.e. replacing one or two words in the headline text in order to introduce humor in the news content. There are about 5,000 distinct news headlines, each having 3 edited versions, thus totaling about 15,000 records.

The data is partitioned into 3 subsets, the Train Dataset with about 64% records, the Dev Dataset with about 16% records, and the Test Dataset with 20% of the records. All edited versions of the same headline are in the same subset (Hossain et al., 2020).

One record of the training dataset is comprised of five fields, as shown in Table 1 below.

id	original	edit	grades	meanGrade
12101	Iran successfully launches satellite-carrying rocket into <space/>	tree	32210	1.6

Table 1: Sample Training Data

In Table 1, "id" denotes the unique identifier of an edited headline, "original" refers to the unedited news headline with the words to be replaced identified by the </>tag, "grades" is the concatenation of the rating given to the news headline by each judge and "meanGrade" is the mean of the ratings given to the news headline by all the judges.

3.2 Data Pre-processing

For the data preprocessing part, we first clean the text by removing special characters, converting to lowercase, expanding contractions like won't, can't, etc., and removing stopwords using NLTK. Next, we use the given replacement word and perform the micro-edit in the original news headline sentence. Micro-edits mean replacing the word inside the angular brackets in the original news headline with the given replacement word to get the edited news headline. We do this using regular expressions and we thus create two columns, one for each version of the news headline.

3.3 Creation of Word Embeddings

The next step in preprocessing is the conversion of textual information into numeric vectors prior to feeding the sentences into the models. For this, we use three approaches to create numerical vectors. First is the simple bag of words (BoW) method (Zhang et al., 2010) where we use the count of each token to assign it a vector value. The second technique is the TF-IDF method, which takes into account the importance of the word in a sentence (Ramos et al., 2003). Therefore, we expect there will be some improvements with using TF-IDF over the BoW features. The third technique we used was using the Word2vec embeddings (Mikolov et al., 2013). These embeddings are pretrained on the Google news dataset and they consider semantic information and context while assigning numerical vectors to words. We expect that Word2vec will further improve the performance as it gives a more meaningful representation to words as compared to TF-IDF which ignores semantics. To obtain the vector representation of a complete sentence, we average together all the word-vectors for the words in the sentence. Another important point to note is that since we are dealing with a sentence-pair regression problem, we have to create vector representation for two sentences, first the original news headline and second the edited news headline. Therefore, we combine these vector representations by stacking the individual sentences' numeric arrays in sequence horizontally. These combined vectors are then fed to the machine learning models.

3.4 Model Building

For the model building step, we first create the baseline model. The baseline model is the model which calculates the mean funniness grade from all records in the training set. Now it will always give the prediction as this mean value. After creating this baseline, we then move to build more intelligent models. We experiment with a lot of different models like Support Vector Regressor, Random Forest Regressor, AdaBoost Regressor, XGBoost Regressor, etc. We build these models on each of the three types of vectorized representations of our text which we described above. We also try hyperparameter tuning for each of the models. We select the best-performing models and then even try a Stacking Regressor Model on top of these tuned models. Finally, we evaluate the performance of all these models on the unseen test data.

4 Results

In this section, we present the results obtained from our experiments.

Model	Features	RMSE
Baseline	-	0.5747
Linear-SVR	BoW	0.5744
Linear-SVR	TF-IDF	0.5723
Linear-SVR	Word2Vec	0.5655
NuSVR	BoW	0.5671
NuSVR	TF-IDF	0.5620
NuSVR	Word2Vec	0.5579
SVR-RBF	BoW	0.5684
SVR-RBF	TF-IDF	0.5598
SVR-RBF	Word2Vec	0.5582
Random Forest	BoW	0.5879
Random Forest	TF-IDF	0.5957
Random Forest	Word2Vec	0.5652
AdaBoost	BoW	0.5684
AdaBoost	TF-IDF	0.5776
AdaBoost	Word2Vec	0.5709
XGBoost	BoW	0.5632
XGBoost	TF-IDF	0.5682
XGBoost	Word2Vec	0.5622
Stacking	BoW	0.5611
Stacking	TF-IDF	0.5640
Stacking	**Word2Vec**	**0.5503**

Table 2: Results on Test Data

As shown in the table above, we are using the metric of Root Mean Squared Error (RMSE) for assessing the performance of our work. The baseline model we build gave an RMSE score of **0.5747** on the unseen test data. We are now comparing

all our models on the value of this metric on test data and the model which can minimize this error by the maximum amount is to be chosen as the best-performing model.

From the results in Table 2, we observe that the Stacking Regressor when trained with Word2Vec embeddings obtained the least overall RMSE among all model and embedding combinations. It resulted in an RMSE of **0.5503** for our task of predicting the funniness level of the edited headline. This regressor is stacked with hyperparameter tuned layers of SVR-RBF, NuSVR, and Linear-SVR models. The possible reason why this model is performing the best out of the lot is that the stacked ensemble models combine the strengths of the individual models and thereby, yield better performance than each of the individual models. Now, the individual models in this stacked model are the best versions of themselves since they have been hyperparameter tuned. Therefore, Stacking Regressor further improves the overall performance as compared to each of the base models. In other words, this model performs well primarily because stacking reduces variance and results in a more robust model by combining the predictions of multiple models.

To get a sense of how well our model works, we compare our results to the official results as reported in Hossain et al. (2020) for Subtask 1. The winner of the task, Hitachi (Morishita et al., 2020) achieved an RMSE score of 0.4972 by using an ensemble of pretrained language models. The performance of our model results in an RMSE score of 0.5503 when tested on the Test set. Our system's performance would be somewhere close to the performance of the 22nd best team: Buhscitu with an RMSE of 0.5511 (Jensen et al., 2020), as reported in Hossain et al. (2020).

5 Discussion

From the results in Table 2, we can see that overall, models built on Word2vec feature vectors are giving a better performance than models built on BoW or TF-IDF. The possible explanation for this observation is that Word2vec retains semantic information and has a sense of contextual information compared to TF-IDF and BoW vectorized features. Additionally, BoW and TF-IDF generate large sparse vectors unlike Word2Vec, where the size of the embedding is small and isn't sparse. The BoW embeddings result in the worst performance because they lose contextual information. The TF-IDF embeddings are better than the simple BoW embeddings because they contain additional information regarding the level of importance of individual words. However, it is not as good as Word2Vec because it still results in a large sparse matrix of embeddings.

Another interesting observation from the results in Table 2 is that variants of SVM, both NuSVR and SVR-RBF performed well overall. This could be explained by the robustness of SVR against outliers and its ability to generalize well. The RBF kernel introduces non-linearity to the SVR model, while NuSVR uses a parameter nu to control the number of support vectors. Therefore, both of these models perform better than the Linear-SVR, a simple model that only uses a linear kernel.

We believe that these results are important because our system sufficiently outperforms the baseline even without using any deep neural network based models as done in other related work. By using machine learning models, our system is more interpretable and it can provide reasonable explanations behind the prediction results which is currently lacking in other solutions present in the literature.

6 Conclusion

In this work, we experimented with various word embedding techniques and with different model architectures to solve the task of predicting rating for the funniness of a one-line headline given it has been micro-edited (one semantic entity exchanged with another) by humans. We found that the minimum RMSE score (0.5503) was achieved by using a Stacking Regressor Model when it was trained with Word2Vec embeddings. This model was stacked with hyperparameter tuned layers of SVR-RBF, NuSVR, and Linear-SVR models. The future work for this project would be to experiment with more pre-trained embeddings or to come up with more feature engineering techniques so as to further improve the performance.

7 Repository URL

The Github repository of our project can be found at

`https://github.com/UOFA-INTRO-NLP-F21/f2021-proj-chiragdaryani`

References

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)*, pages 113–117.

Luke De Oliveira and Alfredo L Rodrigo. 2015. Humor detection in yelp reviews. *Retrieved on December*, 15:2019.

Christian Hempelmann, Victor Raskin, and Katrina E Triezenberg. 2006. Computer, tell me a joke... but please make it funny: Computational humor with ontological semantics. In *Flairs conference*, volume 13, pages 746–751.

Nabil Hossain, John Krumm, and Michael Gamon. 2019. " president vows to cut¡ taxes¿ hair": Dataset and analysis of creative text editing for humorous headlines. *arXiv preprint arXiv:1906.00274*.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. Semeval-2020 task 7: Assessing humor in edited news headlines. *arXiv preprint arXiv:2008.00304*.

Kristian Nørgaard Jensen, Nicolaj Filrup Rasmussen, Thai Wang, Marco Placenti, and Barbara Plank. 2020. Buhscitu at semeval-2020 task 7: Assessing humour in edited news headlines using hand-crafted features and online knowledge bases. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 824–832.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Terufumi Morishita, Gaku Morio, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at semeval-2020 task 7: Stacking at scale with heterogeneous language models for humor recognition. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 791–803.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Julia M Taylor and Lawrence J Mazlack. 2004. Computationally recognizing wordplay in jokes. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.