Check-in Report: Towards an automated system for rating humor content in textual documents

Chirag Daryani
cdaryani@ualberta.ca

Varshini Prakash
vprakash@ualberta.ca

1 Literature Review

We reviewed several papers that would aid us in the implementation of the project. Liu et al. (2018) extracted latent features from humor using Word2Vec to evaluate meaning distance between content word pairs in a sentence. 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 such as Word2Vec , FastText, and GloVe word embeddings. Combination of hyperparameter-tuned variations of these models through an ensemble learner using regression has also proved to be successful in the past.

2 Methods

The task we are solving is a two-sentence 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. The core method we employ for solving this task is experimenting with various machine learning models 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.

For the data preprocessing part, we use regular expressions to perform the micro-edit in the original news headlines given the replacement words and then we create two columns, one for each version of the news headline. We then try cleaning the text by removing special characters, converting to lowercase, expanding contractions like won't, can't, etc., and removing stopwords using NLTK. The next task in preprocessing was the vectorization of textual information before it is fed to the models. For this, we have tried three approaches to create numerical vectors. First is the simple bag of words (BoW) method where we use the count of each token to assign it a vector value. We use bi-grams for this to get some sort of sequential information in our features. The second technique we tried was TF-IDF which considers the importance of the word in a sentence into consideration. The third technique we tried was using the Word2vec embeddings pretrained on the google news dataset which incorporates semantic information while assigning numerical vectors to words. We then combined the vector representation of both sentences into a single vector which was fed to the models.

For the model building step, we first create the baseline model which always predicts every rating as the overall mean funniness grade in the training set. We then move to building more intelligent models like linear regression, support vector regressor, random forest regressor, etc. on each of the three types of vectorized data described above. We also try hyperparameter tuning for each of the models. Finally, we evaluate the performance of these models on the unseen test data.

3 Evaluation Protocol

For assessing the performance of our work, we are using the metric of Root Mean Squared Error (RMSE). The baseline model we build gave an RMSE score of **0.57471** on the unseen test data. We are 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 will be chosen as the best-performing model.

4 Results

As described above, we tried various machine learning models on each of the three techniques of text vectorization. We observed that as expected, models built on Word2Vec features gave better performance than models built on BoW or TF-IDF vectorized features. This is so as Word2Vec does not discard the semantic relationships between words while assigning them numerical vectors and this information is useful for our models. After building these models, we also did some hyperparameter tuning on the models to increase the performance. While comparing the different model architectures, our initial results show that SVM models are performing the best, especially the NuSVR and RBF-kernel-based SVRs. The table below shows some of the results we have observed till now.

Model	Features	RMSE ↓
Baseline	-	0.57471
SVR-RBF	BoW	0.56848
AdaBoost	BoW	0.56842
Linear Reg.	Word2Vec	0.56748
NuSVR	BoW	0.56716
Linear-SVR	Word2Vec	0.56559
Random Forest	Word2Vec	0.56523
NuSVR	TF-IDF	0.56205
SVR-RBF	TF-IDF	0.55981
SVR-RBF	Word2Vec	0.55829
NuSVR	Word2Vec	0.55799

Table 1: Results on Test Data

These are not the final results and we will try further hyperparameter tuning of these models and also try other model architectures in order to improve these results in the coming weeks.

5 Project Completion Plan

The following are the main tasks we need to complete as a part of the project along with their respective time estimates:

Task	Time Estimate
Try more data preprocessing techniques	2 days
Experiment with different word embeddings: Word2Vec, FastText, ELMO etc.	3 days
Testing with simple neural Networks, LSTM	2 days
Testing with an ensemble model	2 days
Drafting the methods section of the report	3 days
Drafting the discussion section of the report	2 days
Drafting the results section of the report	1 day
Refining the final report based on feedback	1 day

Table 2: Project Completion Plan

6 Repository URL

The Github repository of our project can be found at

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

References

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*.

Lizhen Liu, Donghai Zhang, and Wei Song. 2018. Exploiting syntactic structures for humor recognition. In *Proceedings of the 27th international conference on computational linguistics*, pages 1875–1883.

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.