Recognizing Humor and Predicting Humor Ratings in Short Texts

Daniel Berg Thomsen, Jean-Baptiste de la Broïse, Eelis Mielonen
École Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*—The task of automatically recognizing and assessing humor in text is a relatively new task, which has seen many improvements with the advent of powerful neural network architectures. In our work we start out by evaluating a set of input representations with many different models, among which is a transformer architecture (BERT) fine-tuned on the SemEval 2021 task 7 data set. We use the humor labels and ratings to train a set of models which can classify texts as humor, and others which predict the humor rating of the given text. We demonstrate that fine-tuning BERT for our task outperforms all other approaches in terms of $F1$-score and classification accuracy when recognizing humor in the texts, and feeding BERT encodings to an Elastic Net regressor leads to the best results in terms of RMSE and R^2 in the case of predicting the humor ratings.**

I. Introduction

Humor detection is the task of detecting whether a given text document is humorous or not. Given that there is no known mathematical relation between the structure of language and humor, the humor associated with a given document is by definition provided by a set of human annotators. Humor recognition is a very interesting task as it was considered very difficult until very recently. The introduction of new neural network architectures and the use of neural embedding allows us to get outstanding results that would not have been possible 10 years ago. Humor recognition has many applications - one of which would be in the entertainment industry. If there is a model which reliably predicts the level of humor in a given text, one could (for example) validate and improve upon transcripts written with the intention to be humorous.

In this paper we attempt to train models which produce a slightly different prediction for a given text document, the first of which being whether the given document was *intended* to be humorous, and the latter of which being the average humor rating a set of human annotators produced for the document. The distinction between whether a text is humorous or if the *intent* is to be humorous is important, as a given text can be inadvertently humorous (i.e. not meant to be humorous, but humorous regardless).

The problem can be formulated as a supervised learning task, where each input is a short document paired with labels representing whether the given document was intended to be humorous or not, in addition to the average humor rating provided by the annotators.

The data set in our work is distributed as part of ”The 15th International Workshop on Semantic Evaluation” (SemEval-2021) and was produced by asking a set of human annotators the following questions [1]:

- Is the intention of this text to be humorous?
- (If it is intended to be humorous) How humorous do you find it? (1-5)

The annotators were also asked a set of questions related to the controversy of the given jokes, but since the project proposal offered by the Human-Computer Interaction group did not deal with any tasks related to the controversy we have neglected to mention them here.

The jokes themselves can be multiple sentences long. The following is an example from the data set:

> *I never finish anything. I have a black belt in partial arts.*

We make use of scikit-learn [2] and PyTorch [3] to implement our models. For pre-processing we make use of NLTK [4], Gensim [5], and spaCy [6]. The source code for our work is available as a code repository on GitHub.

II. Related work

Early work on recognizing humor in short documents made use of style-based input representation, and then trained classifiers on this representation [7]. The style-based features of this work were inspired by linguistic theories of humor, whereby the presence of alliterations, antonyms and adult slang are suggested as good features. This work also presents a commonly used data set for the field, namely the ”one-liners” data set.

Further attempts to create feature representations have been made in order to increase the accuracy of a humor recognition model. In some work it was found that a simple Word2Vec representation of the documents achieved very good results on the one-liners data set [8].

Other work in the area focuses on using neural embeddings produced by transformer architectures, where the authors compare their approach with a convolutional neural network (CNN) architecture [9]. They find that using the transformer architecture is superior in the task of recognizing humor on multiple data sets.

III. Models and Methods

To solve this task we implement a wide array of different models and input representations, taking inspiration from previous work in the field. Each of the models tested have their respective hyperparameters selected using grid search.

To evaluate the performance of each configuration (when searching for the best hyperparameters) we perform 5-fold cross-validation. In the end, we provide the $F1$-score and classification accuracy computed on a testing set containing samples not seen during training. For the task of predicting the average humor ratings we provide the root-mean-squared-error (RMSE) and coefficient of determination (R^2) computed on the testing set.

A. *Vector Transformations*

The first approach attempted in our work was to represent the data in five different ways, and then to test a set of models to see if there was a certain representation which worked particularly well with a given model.

As a baseline for our other feature representations we implemented the set of features from the work of Mihalcea and Strapparava [7]. Even though the work of the authors indicates that better results can be achieved with content-based representations, these features are still of interest since they are easily interpreted. In addition to considering the antonyms, alliterations and adult slang of each document, we also consider the presence of synonyms. The choice to use synonyms was based on the intuitive observation that many of the jokes contained a repetition of words with the same meaning.

These features were computed in the following way:

- Antonyms/synonyms: For each word in the document, we count the amount of antonyms/synonyms for that word in the rest of the sentence. To find a list of antonyms/synonyms a dictionary is required. We used WordNet [10] as the dictionary in our work.
- Alliterations: For each word in the document we compute the phonemes using a dictionary. We used the Carnegie Mellon University pronouncing dictionary. Then we iterate over the words in the document, adding the first phoneme of the word to another dictionary each time. On each word we also check if the first phoneme is already present in the dictionary, and increment a counter variable each time a match has occurred.
- Adult slang: We iterate over each word in the document, and check if that word could be considered "adult slang" using a hierarchy of domain associations for words. We used the WordNet Domains hierarchy for these purposes [11].

Additionally, we decided to use the term frequency-inverse document frequency (tf-idf) statistic [12], Word2Vec [13] and embeddings produced implicitly by a transformer architecture (BERT) to embed the given documents. The reason for including tf-idf in our study is that (to the knowledge of the authors) there have been no previous attempts to use this representation for this task in the literature. Word2Vec (without added hand-crafted features) has been shown to yield good performance on this task [8]. Additionally, it seems that neural embeddings produced by transformer architectures show good promise in comparison to using a CNN architecture.

B. *Transfer Learning with BERT*

BERT (Bidirectional Encoder Representations from Transformers) is a language representation model that was developed by Google as a so-called fine-tuning approach [14]. The fine tuning approach can be divided into two distinct phases: the pre-training phase and the fine-tuning phase. During pre-training BERT is trained on a large corpus to produce general language representations. After this stage, the model can be specialized to perform a broad range of language modeling tasks by adding a new layer on top of the output layers of the pre-trained model, and training this composite model on a task-specific data set.

Fine tuning can be approached in multiple ways. One of the key factors to consider when choosing the approach is the large size of the model (the base model used in this study has 110 million parameters). One can choose to make updates to all the layers in BERT, which is costly, or simply choose to only train a smaller subset from the last layers. In our study, we experimented with two such approaches. Firstly, we only trained the weights of the added layer, or in other words, used the pre-trained BERT as an encoder. Secondly, we retrained the top 5 layers of BERT along with the output layer (freezing all the other layers). To this end, we used the HuggingFace library [15], which provides a wide variety of pre-trained transformer arcitectures (including RoBERTa and GPT-2) as well as PyTorch optimizers for them. When fine tuning BERT by retraining the last 5 layers and the added top layers, we used an Adam optimizer with a learning rate of $2e-5$, a batch size of 16, and carried out the training for 4 epochs over a training set of 7000 samples. The choices for these parameters are based on the recommendations of the BERT developers. The Adam optimizer also has additional parameters, which we left to their default values: $\beta_0 = 0.9, \beta_1 = 0.999, \epsilon = 10^{-7}$.

IV. Experiments

To make use of Word2Vec representations of documents as inputs to our models we tried the following approaches:

- Averaging the vectors computed for each word of a document
- Computing the tf-idf-weighted average of the vectors computed for each word of a document

The reason for computing the tf-idf-weighted average is to give more weight to rarer words. We postulated that this might capture unexpected or unique word choices in jokes.

For both of these approaches we observed clustering structure by creating visualizations of the t-distributed stochastic neighbor embeddings (t-SNE) for the resulting inputs which are available in Figure 1 and 2.

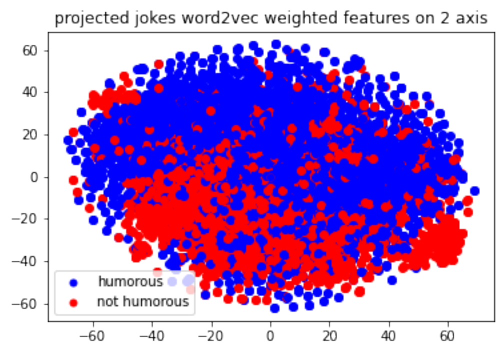

Figure 2. Visualization for tf-idf-weighted averaged vectors from Word2Vec, using t-SNE, perplexity=25. It would seem as though two overlapping clusters can be seen using this input representation.

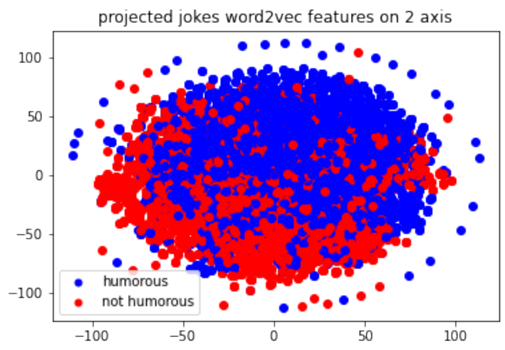

Figure 1. Visualization for averaged vectors from Word2Vec, using t-SNE, perplexity=25. Two overlapping clusters can be seen using this input representation.

A different set of models were used for the two different tasks of this paper.

1) Classifiers: For the classification task we decided to use the following set of models, tuning the following hyperparameters[1] using grid search (the exact hyperparameters can be found in the Appendix VII):

- Multinomial Naïve Bayes: the smoothing prior α
- Support-vector machine (using linear kernel): regularization strength C (we use the dual method for tf-idf features as the size of the vocabulary is greater than the size of the data set).
- Gradient boosting machine: number of estimators, maximum depth of tree, minimum child weight and loss reduction γ required to add a partition on a leaf node.
- Neural network with varying hidden layer dimensions.

We consider the first two of the models above to be baseline models, as they were the ones used in the paper by Mihalcea and Strapparava. We chose to include gradient boosting machines since it is an interpretable model with the ability to fit a relatively complicated function. The neural network was added because of its capability to fit a highly complicated function.

2) Regressors: For the regression task we used the following models. Hyperparameters are detailed like in the previous section:

- Elastic Net: regularization strength α and $l1$-ratio.
- Gradient boosting machine: number of estimators, maximum depth of tree, minimum child weight, learning rate α and loss reduction γ required to add a partition on a leaf node.
- Neural network with one hidden layer: the number of neurons in the hidden layer.

We chose elastic net as our baseline model over ridge or lasso regression, since elastic net is a mixture of the two approaches.

V. Results

A. Classification

We obtain good results on the classification task with word embeddings (see Table I and II). Overall, it seems like it is possible to achieve good results with every input representation (except the style-based features) if the right model is paired with it. Out of all these feature representations, the best results were still had when using BERT encodings. When using these encodings, the difference in performance between using a neural network and gradient boosting machine is negligible. Fine-tuning BERT by freezing all except the last 5 layers resulted in an accuracy of 0.927 (average over 8 different splits of the data), and an F1-score of 0.941. This is the highest accuracy out of all the implemented methods.

Table I
ACCURACY FOR DIFFERENT FEATURE REPRESENTATIONS AND MODELS

	multinomial_nb	**linear_SVM**	**xgboost**	**neural_net**
tfidf	0.842500	0.826875	0.778125	0.83625
w2v_avg	0.641875	0.840625	0.831875	0.85375
w2v_weighted	0.683750	0.824375	0.833125	0.85250
style	0.615000	0.616250	0.621250	0.61500
bert_embeddings	0.864375	0.894375	0.917500	0.92000
fine tuned bert	NA	NA	NA	0.92700

Table II
F1-SCORE FOR DIFFERENT FEATURE REPRESENTATIONS AND MODELS

	multinomial_nb	**linear_SVM**	**xgboost**	**neural_net**
tfidf	0.880342	0.860594	0.818600	0.868077
w2v_avg	0.773964	0.873071	0.867029	0.884044
w2v_weighted	0.793974	0.859850	0.870325	0.881407
style	0.761610	0.761831	0.760664	0.761610
bert_embeddings	0.894915	0.914947	0.933668	0.933054
fine tuned bert	NA	NA	NA	0.940283

[1]the hyperparameter search for models using tf-idf was limited because of computational restrictions resulting from our chosen vocabulary size

B. Regression

The RMSE and R^2 computed on the testing sets for the regression task can be found in Table III and IV. Among these we make note of the fact that the BERT encodings again outperform other input representations, and the best results are achieved when fine tuning BERT. The RMSE of the elastic net is on-par with the fine-tuned BERT model, however it had a slightly higher R^2 statistic.

Table III
RMSE FOR DIFFERENT FEATURE REPRESENTATIONS AND REGRESSORS.

	elastic_net	**xgboost**	**neural_net**
style	0.574321	0.578341	0.574252
tfidf	0.550051	0.559480	0.688011
w2v_avg	0.535779	0.571204	0.652232
w2v_weighted	0.538551	0.560144	0.567671
bert_embeddings	0.517665	0.568101	0.548361
fine-tuned bert	NA	NA	0.509523

Table IV
R^2 FOR DIFFERENT FEATURE REPRESENTATIONS AND REGRESSORS

	elastic_net	**xgboost**	**neural_net**
style	0.002835	-0.011172	0.003077
tfidf	0.085330	0.053704	-0.431029
w2v_avg	0.132182	0.013630	-0.286063
w2v_weighted	0.123178	0.051459	0.025793
bert_embeddings	0.189869	0.024318	0.090943
fine-tuned bert	NA	NA	0.149521

VI. DISCUSSIONS

We decided to try many different models in our work. As a result we were not able to perform a very in-depth search of hyperparameters for each model. This, however, allowed us to compare the different models for the different feature representations which is valuable as it allows us to know in which direction to head in if we want to get better results using those models.

The style-based features were initially designed to work for a slightly different format than the jokes of the data set that we have been working on. Concretely: alliteration might be a more important element of jokes constrained to a single line, whereas longer jokes like some of those found in our data set might use more complex stylistic elements to incorporate humor. Looking at the jokes of our data set we could identify many anaphora, but very few jokes containing alliterations. Despite this slightly different situation we did indeed observe performance similar to that in the work of previous studies making use of these features on their own.

VII. CONCLUSION

In this paper we have shown that it is possible to infer humorous intent and humor ratings of short documents using properly engineered feature representations and machine learning models. We also showed that using neural encodings produced by a transformer architecture was superior to using style-based features, as well as some common representations found within the field of natural language processing such as tf-idf and Word2Vec. Finally, we also observed that fine-tuning the last couple of layers of BERT with the output layers lead to incremental improvements in performance over just training the output layer on these tasks.

Though this is not the first work on humor prediction using various approaches (including transformer architectures), the previous attempts have dealt with other data sets and usually do not include the task of predicting the humor rating of a given document.

APPENDIX

VIII. HYPERPARAMETERS IN GRID SEARCH

A. Classifiers

- Multinomial Naïve Bayes:
  $\alpha =$ `np.logspace(-1, -6, num=5, base=4, endpoint=False)`
- Support-vector machine (using linear kernel):
  $C =$ `np.logspace(1, 10, num=6)`
- Gradient boosting machine:
  number of estimators $= [10, 50]$
  maximum depth of tree =`range(3,8,2)`
  minimum child weight = =`range(1,6,2)`
  $\gamma = [10^{-i}$ `for` i `in range(2,6)`$]$ (this is not adjusted when using tf-idf as the input embedding)
- Neural network with one hidden layer:
  hidden layer dimensions $= [(10, 10), (50,)]$
  $\alpha = [10^{-i}$ `for` i `in range(2,5)`$]$

B. Regressors

- Elastic Net:
  $\alpha =$ `np.logspace(1, -4, num=4)`
  $l1$-ratio $=$ `np.linspace(0.1, 1, num=3)`
- Gradient boosting machine:
  number of estimators $= [10, 100, 1000]$ (this is the only hyperparameter adjusted when using tf-idf)
  maximum depth of tree $= [2, 5, 20]$
  minimum child weight $= [1, 2, 5]$,
  $\alpha =$ `np.logspace(0, -4, num=1)`
  $\gamma =$ `np.logspace(-1, -5, num=4)`
- Neural network with one hidden layer:
  hidden layer dimensions $= [(10,), (100,), (500,)]$.
  hidden layer dimensions (when using tf-idf) $= [(10,), (50,)]$.

References

[1] J. A. Meaney, S. Wilson, and M. W. Chiruzzo L., "Semeval-2021 task 7: Hahackathon: Incorporating demographic factors into shared humor tasks." In Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval 2021), 2021.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[4] E. Loper and S. Bird, "Nltk: the natural language toolkit," *CoRR*, vol. cs.CL/0205028, 07 2002.

[5] R. Rehurek and P. Sojka, "Gensim–python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[6] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, to appear.

[7] R. Mihalcea and C. Strapparava, "Making computers laugh: Investigations in automatic humor recognition," pp. 531–538, Oct. 2005. [Online]. Available: https://www.aclweb.org/anthology/H05-1067

[8] D. Yang, A. Lavie, C. Dyer, and E. Hovy, "Humor recognition and humor anchor extraction," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 2367–2376. [Online]. Available: https://www.aclweb.org/anthology/D15-1284

[9] O. Weller and K. Seppi, "Humor detection: A transformer gets the last laugh," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3621–3625. [Online]. Available: https://www.aclweb.org/anthology/D19-1372

[10] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[11] B. Magnini and G. Cavaglià, "Integrating subject field codes into WordNet," in *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*. Athens, Greece: European Language Resources Association (ELRA), May 2000. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2000/pdf/219.pdf

[12] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[15] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6