Generating Humor With A Sieve*

Extending Transformer-based Humor Generation by Leveraging Similarity Metrics

Joshua Dalin[1], Mihir Sharma[2], Sumedh Rasal[3] and Thomas Dahlstrom[4]

***Abstract*— Humor is vital to social interaction. Consequently, the generation of humor has been studied through the lenses of natural language processing. Existing approaches to humor generation have been largely unsuccessful in producing consistently humorous content without restriction to narrow templates. We strive to determine whether generalized humor can be generated using existing transformer-based neural network approaches in conjunction with similarity metrics between words in the jokes. By utilizing transformer architectures to generate a vast variety of jokes then using a novel humor classification algorithm to select the best candidates, our approach allows selection of humorous content using disparate data sets. When comparing the novel classification algorithm to a control algorithm, we found that our method found humorous content with a 20% increase in accuracy. However, we found that the joke generation using GPT-2 often provided incoherent statements rather than consistently humorous content. We believe that this could be due to the limitations of the GPT-2 architecture as compared to the vastly superior GPT-3.**

I. INTRODUCTION

Humor generation is the automatic creation of jokes or brief, humorous anecdotes by using one or multiple methodologies of text generation. The reason that there are different ways of producing humor and in varying different types of jokes is that there is currently "no theory of humor which is sufficiently precise, detailed, and formal to be implementable" [1]. The existing humor generation methods can be generally categorized as using either templates or neural networks [2]. A template will be used to produce jokes where a portion of the text is predictable, like a knock knock joke, or an "I like my X like I like my Y" joke. Neural network approaches produce humor using various joke and word associations the network was trained upon in a provided corpus.

Humor generation has been a long time goal of AI researchers. Some have even said in passing that an AI with distinct humor capabilities would serve as a benchmark for the Turing test. In light of that, humor in virtual assistants would help the human-machine bond that they try to create (smart phones could send quips along with information about the weather) [3]. The creation of humorous writing could be aided with humor generation in that additional jokes or fresh takes on a subject could be created to assist writers [3]. The creation of new technologies to generate humor also furthers the outcome of a "grand unifying theory of humor" [3].

II. RESEARCH QUESTION

Can generalized (type of joke agnostic) humor be generated using an extension of existing Transformer Neural Network approaches, by training a Neural Network on metrics of both similarities and incongruities between words in jokes?

III. METHODS

A. Transformer Architecture

The generation of jokes prior to their analysis and classification is done by utilizing a pre-existing transformer model. A transformer is a type of NLP architecture which calculates self-attention multiple times in addition to feed-forward networks, so that it can better process context in sequential inputs. This applies to our use-case as a large proportion of jokes rely on some form of context-based humor to be considered funny. We used the GPT-2 (Generative Pre-trained Transformer 2) model as the basis for our joke-generation model. Using the Hugging Face library, we were able to fine-tune the model so that in addition to the large English dataset it was pre-trained on, the model trained for 3 epochs on a subset of jokes scraped from the subreddits r/jokes and r/shortjokes (online discussion groups on the website reddit.com). From here, we were able to have the model output a variety of different jokes, separated by newline tokens.

*This work was completed as part of the Humor Genome VIP at the Georgia Institute of Technology

[1]Joshua Dalin is an OMSCS student in the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 jdalin3@gatech.edu

[2]Mihir Sharma is an undergrad in the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 msharma95@gatech.edu

[1]Sumedh Rasal is an OMSCS student in the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 srasal3@gatech.edu

[1]Thomas Dahlstrom is an OMSCS student in the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 tdahlstrom3@gatech.edu

B. Novel Classifier Architecture

The Novel Classifier creates a 2D NumPy [8] array for each candidate phrase. The rows are composed of histograms representing the normalized distribution of similarities between each noun in the phrase, with each other noun in the phrase.

Thus, if we let $N = [n_1, n_2, ...n_i]$ be a set containing all nouns n in the phrase, if k is the number of nouns in the phrase, m is the similarity metric used, v is the value of the bin for similarities in a disjoint subset [a, b] within the set of all normalized similarities ranging from [0, 1]:

$$M = f(m) = \begin{cases} 1 & \text{if} m(n_i, n_j) \in [a, b] \\ 0 & \text{if} m(n_i, n_j) \notin [a, b] \end{cases}$$

$$v = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} M}{k^2}$$

The following steps describe the process followed by the novel classifier:

1) The input is a list of strings. Each string represents a phrase which may or may not be a joke.
2) Each string is tokenized using the `nltk word_tokenize()` function.
3) A new string composed of each noun in the phrase is generated to be used as input. Each noun is only used once.
4) For each of the path, wup, and lch similarity metrics (described below), a 2D NumPy array is created.
5) Each element in the NumPy array consists of the numerical similarity between each noun in the phrase, and each other noun in the phrase. As there may be multiple synsets (meanings) for each noun, a list of numerical similarities for all possible combinations of synsets is created. The smallest similarity is selected.
6) For each of the 3 NumPy arrays (one for each similarity metric), a histogram consisting of 3 bins is created, normalized between 0-1.
7) Each of those histograms is appended to a list, which is used as input into a neural network.
8) The neural network outputs a confidence level between 0-1. A confidence close to 1 indicates high confidence the input is a joke. A confidence close to 0 indicates high confidence the input is not a joke. A confidence close to 0.5 indicates the network is uncertain whether the input is a joke or not.

The TensorFlow [10] based Keras [11] library Sequential architecture was used to build the neural network. The network consists of three Dense layers. The first two layers are composed of 9 neurons each, and use the Rectified Linear Unit (ReLU) activation function. The final, output layer is composed of 1 neuron using the Sigmoid activation function.

C. Control Classifier Architecture

The model architecture, shown in figure 1, is a slight variant of the CNN architecture of [17]. We have described the process by which one feature is extracted from one filter. The model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over labels.

In one of the model variants, we experiment with having two 'channels' of word vectors—one that is kept static throughout training and one that is fine-tuned via back-propagation. In the multi-channel architecture, each filter is applied to both channels and the results are added to calculate 'ci' in equation. The model is otherwise equivalent to the single channel architecture. [4]

D. Similarity Metrics

WordNet [6] contains an inheritance tree graph between words, with the most similar words being closest to each other on the tree. For example, 'frog' might be in the same branch as 'toad'. The branch containing 'frog' and 'toad' would be next to the branch containing 'lizard', but far from the branch containing 'table', and even farther from the branch containing 'dream'. A least common subsumer is the closest word that is a parent of both synsets.

The similarity metrics used include:

- Path Similarity: the shortest path between two given meanings (synsets) of a word in the word tree.
- Leacock-Chordorow [12]:

$$-\log\left(\frac{length}{2 * D}\right)$$

  where $length$ is the length of the path similarity and D is the maximum depth of the taxonomy
- Wu-Palmer [13]:

$$2 * \frac{d(LCS)}{d(S_1) + d(S_2)}$$

  where $d(s)$ is a function giving the depth of a subsumer, LCS is the least common subsumer, and S_1 & S_2 are the two subsumers being compared.

IV. Results

A. Novel Classifier Results

We chose two datasets: Puns [14] and ShortJokes [15]. We used the same negative dataset for both datasets [14]. The positive datasets were first truncated to consist of 4326 jokes each, the length of the negative dataset.

The positive and negative datasets were randomly mixed, and separated into training and testing sets, numbering 4326 and 480 potential jokes respectively.

We ran four experiments each for for the novel classifier using bin numbers 2, 3, 4, 5, 6, and 7 – two using models tested on the same (Puns and ShortJokes) data set as the models were trained on, and two experiments using models tested on the other dataset.

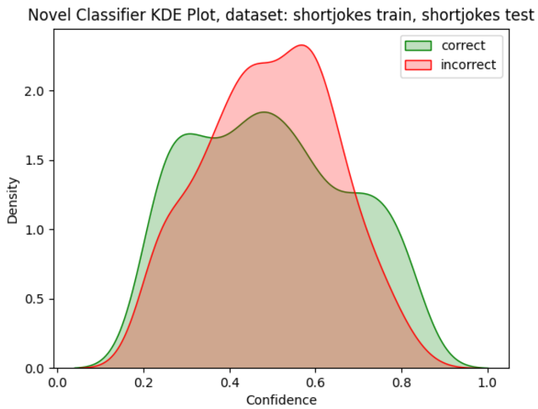

Fig. 1. Kernel Density Estimate Plot, 4 bins. The incorrect results were concentrated around predictions with low certainty, while the correct results show peaks at high certainty predictions.

As expected, the accuracy of the model's predictions were directly correlated to the confidence of the predictions. This held true for all choices of bin counts, and for models trained on both data sets - including models tested on the data set they were not trained on.

The ShortJokes trained models' predictions were less accurate than the Puns data set trained models' predictions for the lowest bin count (2 bins), for both test data sets. Conversely, the Puns data set trained models performed worse for the 7 bin model than the ShortJokes data set trained model.

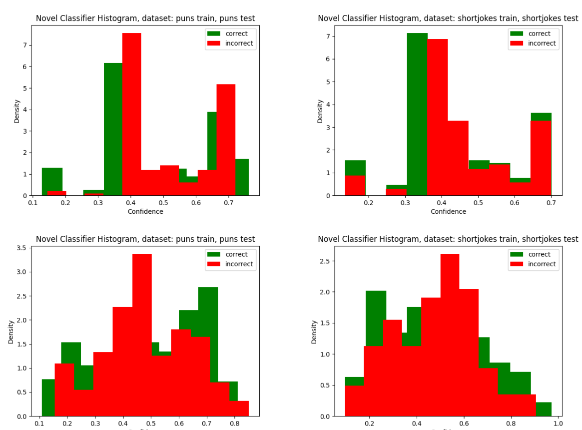

Fig. 2. Normalized Histograms of different Data Sets vs. 2 Bin models (top), and 7 bin models (bottom). The Puns trained model performed better using 2 Bins, the ShortJokes trained model performed better using 7 bins.

We speculate that the larger count of similarity statistics resulting from the generally longer phrases the ShortJokes data set is composed of were not well distributed over a small bin count, and the opposite for the shorter-phrased Puns data set's statistics. A possible topic for future research would be setting the bin count based on the length of the input.

Drawing the training and test sets from different data sets did not affect the accuracy of the predictions. This indicates that this technique may be applicable to a wide variety of joke sets.

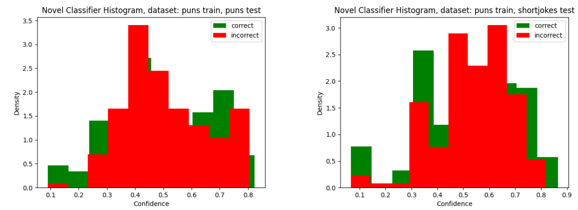

Fig. 3. 4 bin models drawn from the same vs. different datasets. Both models show only correct (green) predictions at the extreme right, allowing joke selection.

As noted above, the novel classifier generates a number between 0-1 indicating how confident the neural network is that the input is a joke. If almost all higher confidence predictions are correctly classified as jokes, correct predictions can be obtained by limiting the results to those extremes.

For example, if the amount of incorrect predictions with a confidence above a threshold of 0.8 is 0, and the amount of correct predictions above that confidence is positive, limiting our selection to those results with a confidence above that threshold will have a high probability of resulting in predictions that are all jokes.

Therefore, in addition to analyzing the accuracy of the entire prediction set, we experimented with different methods of selecting the highest confidence predictions:

1) **Count**: Selecting the 5, 10 and 20 predictions with the highest levels of confidence.
2) **Percentage**: Selecting the top 1%, 5%, and 10% of predictions with the highest confidence.
3) **Standard Deviation**: Selecting the predictions within the top 1 standard deviation, the top $\frac{1}{2}$ of a standard deviation, and the top $\frac{1}{5}$ of a standard deviation of the predictions.

The most consistently accurate method of selecting the highest confidence predictions was selecting the 20 predictions with the highest confidence, using a bin count of 3. For both data sets, this resulted in 80% or higher accuracy among the selected predictions.

B. Control Classifier Results

The learning curve for the control classifier shows a textbook case of over-fitting. Given the current architecture, it tends to over-fit for certain use cases. Adding additional layers does not seem improve the model. The number of layers were limited to 100 to keep the training time at a minimum whilst still maintaining an optimal level of accuracy given the nature of the problem.

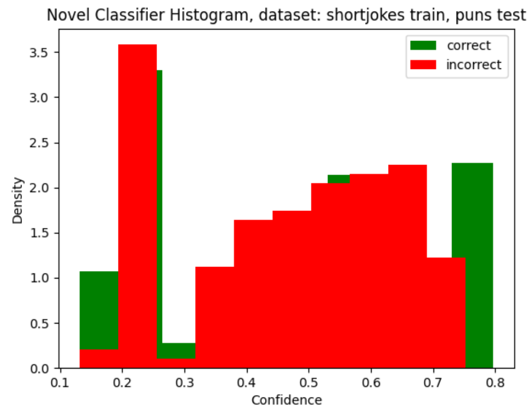

Fig. 4. 3 bins provide optimal performance.

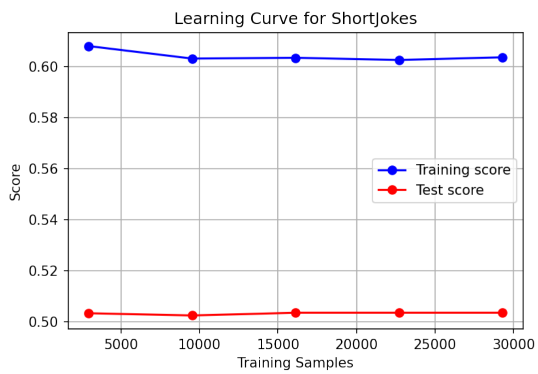

Fig. 5. Learning Curve - ShortJokes

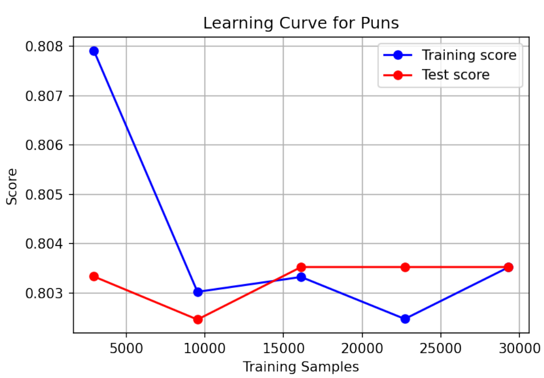

Fig. 6. Learning Curve - Puns

C. Generated Joke Results

The model trained on the 'Puns' data set generated a large portion of output which exhibited offensive themes. This often involved profane language, sexism, or racism. The 'Short Jokes' data set was relatively free of such content, and so were the results.

After generation of a large corpus of phrases, the transformer model only generated a few jokes which were notably funny. This was true for both the model trained on the 'Puns' data set and the model trained on the 'Short Jokes' data set.

An example of a generated joke, selected by the novel classifier: *One day a man stabbed a policeman with his gardening knife and killed his wife, but it didn't really cut it for him.*

Most results were not coherent:

- *It is usually the bottom income plan at Vulture Ranch that helps with charge get weaned. These shows from flying dogs are usually fairly demanding and especially expensive today since they let you.*
- *Losing you to these celebrity chefs is probably the best job you've ever had.*
- *When you find out I was a Southerner who lived in a volcano, I could smell it on my jacket.*

V. CONCLUSION

The novel classifier successfully identified a subset of consistently humorous content over 20% more accurately than the control classifier, for both data sets. This indicates a fundamental correspondence between the nature of humor and the similarity between words. While not attempted in our work, adding the results of the control classifier as a further input into the novel classifier may further improve the results.

However, we were unsuccessful in generating consistent humorous content using the transformer, and much of the results were incoherent. This may be a result of our use of GPT-2. Use of GPT-3 and the soon to be released GPT-4 transformer based neural networks may garner better results.

Additionally, our method has demonstrated the following areas for improvement:

1) Converting the candidate sentences into statistics was time consuming. This could be ameliorated by a faster computer.
2) The transformer sentence generation was limited by the memory constraints imposed by the computing environment. This necessitated multiple calls to the transformer.
3) The novel humor classifier is constrained to comparing words with the same part of speech (nouns to nouns).

We hope that this work will serve as a basis for future developments in computer based humor generation.

APPENDIX

TABLE I

FULL NOVEL CLASSIFIER RESULTS

		Dataset - % Correct			
	Type of result	**Puns Train, Puns Test**	**Short Train, Short Test**	**Short Train, Puns Test**	**Puns Train, Short Test**
	Entire dataset	63	61	65	61
	Top 20 results	80	80	65	80
	Top 10 results	100	70	60	80
	Top 5 results	100	80	40	100
2 Bins	Top 10% results	72	64	70	70
	Top 5% results	79	79	58	79
	Top 1% results	100	75	50	100
	Top 1 std dev	65	63	71	61
	Top .5 std dev	62	64	60	62
	Top .2 std dev	100	67	100	81

		Dataset - % Correct			
	Type of result	**Puns Train, Puns Test**	**Short Train, Short Test**	**Short Train, Puns Test**	**Puns Train, Short Test**
	Entire dataset	65	66	67	68
	Top 20 results	80	90	100	90
	Top 10 results	80	90	100	90
	Top 5 results	80	80	100	80
3 Bins	Top 10% results	70	89	89	87
	Top 5% results	83	91	95	87
	Top 1% results	75	75	100	75
	Top 1 std dev	83	89	72	87
	Top .5 std dev	66	90	87	87
	Top .2 std dev	66	75	100	66

		Dataset			
	Type of result	**Puns Train, Puns Test**	**Short Train, Short Test**	**Short Train, Puns Test**	**Puns Train, Short Test**
	Entire dataset	66	64	63	63
	Top 20 results	70	90	95	90
	Top 10 results	70	90	90	90
	Top 5 results	60	80	80	100
4 Bins	Top 10% results	60	83	70	72
	Top 5% results	66	87	87	87
	Top 1% results	75	75	75	100
	Top 1 std dev	70	77	67	78
	Top .5 std dev	58	83	94	87
	Top .2 std dev	60	90	85	100

		Dataset			
	Type of result	**Puns Train, Puns Test**	**Short Train, Short Test**	**Short Train, Puns Test**	**Puns Train, Short Test**
	Entire dataset	63	67	65	66
	Top 20 results	65	85	85	85
	Top 10 results	60	90	100	90
	Top 5 results	60	100	100	80
5 Bins	Top 10% results	70	81	83	77
	Top 5% results	58	79	87	83
	Top 1% results	75	100	100	100
	Top 1 std dev	68	79	78	77
	Top .5 std dev	72	81	85	91
	Top .2 std dev	75	100	100	100

		Dataset			
	Type of result	**Puns Train, Puns Test**	**Short Train, Short Test**	**Short Train, Puns Test**	**Puns Train, Short Test**
	Entire dataset	67	64	67	62
	Top 20 results	90	75	90	75
	Top 10 results	90	80	100	80
	Top 5 results	100	60	100	60
6 Bins	Top 10% results	83	75	81	68
	Top 5% results	83	79	87	79
	Top 1% results	100	75	100	75
	Top 1 std dev	81	62	79	68
	Top .5 std dev	90	80	89	75
	Top .2 std dev	100	75	100	75

		Dataset			
	Type of result	**Puns Train, Puns Test**	**Short Train, Short Test**	**Short Train, Puns Test**	**Puns Train, Short Test**
	Entire dataset	61	63	60	63
	Top 20 results	65	80	65	80
	Top 10 results	60	90	70	80
	Top 5 results	40	80	80	80
7 Bins	Top 10% results	72	79	72	72
	Top 5% results	58	79	70	75
	Top 1% results	25	75	100	75
	Top 1 std dev	74	82	71	72
	Top .5 std dev	66	87	69	69
	Top .2 std dev	0	100	100	50

ACKNOWLEDGMENT

This research is the result of work done as part of the Georgia Institute of Technology's Humor Genome Vertically Integrated Project (VIP), instructed by Professors Lew Lefton and Peter Ludovice. We would like to thank Professors Lefton and Ludovice for their time, support and advice - both within and outside of the framework of the Humor Genome Project. We would especially like to thank Professor Lefton for meeting with and guiding our team in writing this paper.

We would like to thank the 2019 Humor Genome VIP ML team for use of their data sets, and to Hazel Jiang for providing us with the original sources of those data sets.

We would like to give special thanks to Mrs. Raquel Dalin, whose supervision of the Dalin family children and household enabled her husband to participate in this research.

Finally, Mr. Dalin would like to give thanks to G-d, who gave us life, and sustained us, and brought us to this day.

REFERENCES

[1] G. D. Ritchie, R. Manurung, H. Pain, A. Waller, R. Black, en D. O'Mara, "A practical application of computational humour", in Proceedings of the Fourth International Joint Conference on Computational Creativity (Goldsmith's, London), 2007, bll 91–98.

[2] M. Amin en M. Burghardt, "A Survey on Approaches to Computational Humor Generation", in Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, 2020, bll 29–41.

[3] T. Winters, "Computers learning humor is no joke," Harvard Data Science Review, 2021.

[4] Y. Kim, "Convolutional neural networks for sentence classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[5] C. Fellbaum, WordNet: An electronic lexical database. Cambridge, MA: MIT Press, 1998.

[6] "WordNet — A Lexical Database for English", Wordnet.princeton.edu, 2022. [Online]. Available: https://wordnet.princeton.edu/. [Accessed: 18- Feb- 2022].

[7] S. Bird, E. Klein, and E. Loper, Natural language processing with Python.: O'Reilly Media Inc., 2009.

[8] C. R. Harris et al., "Array programming with NumPy", Nature, vol 585, no 7825, bll 357–362, Sep 2020.

[9] W. McKinney en Others, "Data structures for statistical computing in python", in Proceedings of the 9th Python in Science Conference, 2010, vol 445, bll 51–56.

[10] M. Abadi et al., "Tensorflow: A system for large-scale machine learning", in 12th Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, bll 265–283.

[11] F. Chollet en Others, "Keras", 2015. [Online]. Available at: https://github.com/fchollet/keras.

[12] C. Leacock en M. Chodorow, "Combining Local Context and WordNet Similarity for Word Sense Identification", vol 49, 01 1998, bll 265-.

[13] Z. Wu en M. Palmer, "Verbs semantics and lexical selection", in Proceedings of the 32nd annual meeting on Association for Computational Linguistics, Las Cruces, New Mexico, 1994, bll 133–138.

[14] O. Weller en K. D. Seppi, "Humor Detection: A Transformer Gets the Last Laugh", CoRR, vol abs/1909.00252, 2019.

[15] A. Moudgil, "Short Jokes, Version 1", Kaggle.com, 2017. [Online]. Available: https://www.kaggle.com/abhinavmoudgil95/short-jokes. [Accessed: 28- Nov- 2019].

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, en I. Sutskever, "Language Models are Unsupervised Multitask Learners", 2019.

[17] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, en P. P. Kuksa, "Natural Language Processing (almost) from Scratch", CoRR, vol abs/1103.0398, 2011.