

Creating a Bot to Generate Unique Jokes in Text Format

Motivation

Everyone loves to laugh. Laughing not only lightens our mental load, it has also been shown to induce physical changes in our bodies that reduce stress, ease tension, and stimulate organs [1]. Although humor has often been thought of historically as a trait unique to humans and other highly intelligent animals [2] [3], machine learning software might eventually overturn this notion. In fact, machine learning software has already shown promising results in helping children learn and develop their own language skills [4]. Conversely, the fields of artificial intelligence (AI) and neuroscience have long been intimately related [5], at least in part because of the potential for AI systems to help us learn about ourselves and our minds. The current leading theory of humor suggests that having a sense of humor requires an understanding of social context, which might require mastery of sophisticated functions like self-awareness, empathy, spontaneity, and linguistic subtlety [6]. In this way, we thought creating a joke bot would offer us a somewhat unique perspective in analyzing the rudiments of humor, as well as insights into some of the inherent complications with trying to make a computer “think” like a human.

Problem Definition

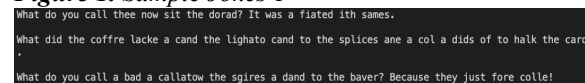
The scope of our problem is significantly larger than any of us understood when initially undertaking this project. Broadly, based on our current understanding, the problem consists of 3 major subproblems, each one dependent upon the completion of the previous subproblem. The first problem is training a model to produce valid text strings of english words. Once the model has established a sufficiently large vocabulary of english words, the next problem is training the model to form coherent sentences from those words. In other words, the model needs to develop a level of understanding of the text it generates. After the model has been trained to output coherent sentences, the final step is training the model to distinguish funny sentences from unfunny sentences, so that the sentence outputs are funny.

Solution Explanation

Initially, we trained a simple, 3-layer recurrent neural network (RNN) to output text based on our dataset of jokes. We used a Python package called textgenrnn to create and train our neural network. To implement textgenrnn is as easy as writing a few lines of code to create a model object with the desired configuration for the model’s output and for the training.

Outputting jokes simply requires displaying that model’s predictions as text. Using textgenrnn allowed us to experiment with several model parameters and evaluate the impact of those parameters on our model’s output. We found that the parameters that most improved our model were using bi-directional encoding (as opposed to not using it) and evaluating tokens as words as opposed to characters. Initially, the model tokenized characters and attempted to generate sentences by predicting the most likely proceeding character for a given character at a certain position in the sentence (i.e., the model determines based on the dataset we used to train it what might be a good character with which to start a sentence, what might be a good character to follow that first letter of the sentence, what might be a good character to follow that second letter, and so on). This resulted in our model outputting a seemingly “random” sequence of words, except for the first few “words” at most. The output consisted almost entirely of English words that seemed entirely disconnected from their surrounding context. For example:

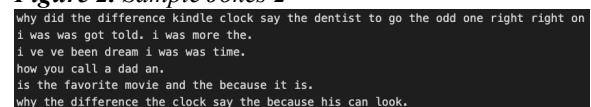
Figure 1. Sample Jokes 1



```
What do you call thee now sit the dorad? It was a fiatid ith sames.
What did the coffre lacke a cand the lighato cand to the splices ane a col a didd of to halk the card.
What do you call a bad a callatow the spires a dand to the baver? Because they just fore colle!
```

After trying different combinations of parameters, this model was able only to generate “sentences” that were incoherent and vaguely resembled common joke structures.

Figure 2. Sample Jokes 2



```
why did the difference kindle clock say the dentist to go the odd one right right on
i was was got told. i was more the.
i ve ve been dream i was was time.
how you call a dad an.
is the favorite movie and the because it is.
why the difference the clock say the because his can look.
```

Based on helpful feedback from our TA, we became aware that our original idea of a solution was simply impractical, and would be unable to produce the results we thought possible when first designing our solution. So we decided to incorporate that knowledge into our project by completely redesigning our proposed solution, and instead implementing a Generative Adversarial Network (GAN). Our reasoning was that GANs provide a solution to all of the same problems we are facing. Namely because GANs make use of a discriminator for improving output of our text-generation model. However, GANs require individual inputs and

outputs: making a GAN implementation quite difficult for processing or outputting text (text which consists of sentences that are composed of words that are composed of characters). There have been only a handful of documented attempts at implementing a GAN for text-generation. We attempted to implement a GAN based on previous successful implementations [7] [8], but were unable to reproduce those results. Ultimately, we were able to use our previous model as a generator, but our attempts at implementing a discriminator to provide useful feedback to our generator were unsuccessful. Our thinking was to teach our discriminator to evaluate whether a joke being output from the generator was funny or not by supplying a list of common english sentences as being “unfunny”, and our dataset of jokes as being “funny”. Our attempts at implementation failed because of our inability to get the two neural nets to learn from the output of their adversary. The generator and discriminator nets are supposed to work in tandem, and we were unable to successfully pass the output of our generator to our discriminator in such a way that our generative model could learn from the discriminator’s feedback. Ultimately, we were unable to produce any output from our GAN aside from the generative output that was already produced by our more naive model that we initially constructed.

Data Description

Our project was run on a subset of a dataset called “short jokes”, which was downloaded from <https://www.kaggle.com/abhinavmoudgil95/short-jokes>. The main problem with this dataset is that many of the jokes contain racial slurs and otherwise inappropriate content. We used grep to search for and remove from the dataset all jokes containing about 100 different words and phrases which we deemed inappropriate or offensive. Leaving in such jokes would result in a joke bot that generates inappropriate content. When we downloaded the dataset from Kaggle, it contained just over 226,000 jokes. After removing all jokes containing the previously mentioned words and phrases, the dataset contained slightly more than 180,000 jokes. This subset of the initial dataset was used to train our model.

The structure of this data is very basic: a csv file containing two columns labeled “ID” and “Joke”. There is no formal “class” label for this data to distinguish between funny and not funny jokes, as the file is made up entirely of jokes (which we assume to be funny). The purpose of this dataset is to teach our model to generate jokes, ideally via the generator neural network in a GAN (see below).

Results

Training

Our model was trained on 20 epochs and it would print sample outputs after every 5 epochs.

Figure 3. Screenshot of Training 1

```
Train for 40 steps
Epoch 1/20
40/40 [=====] - 60s 1s/step - loss: 3.3006
Epoch 2/20
40/40 [=====] - 59s 1s/step - loss: 3.1155
Epoch 3/20
40/40 [=====] - 56s 1s/step - loss: 3.0687
Epoch 4/20
40/40 [=====] - 54s 1s/step - loss: 2.9986
Epoch 5/20
39/40 [=====>.] - ETA: 1s - loss: 2.7291####
```

After the first 5 epochs it starts to produce strings that resemble sentences. At this point our model is still learning and only has a limited vocabulary to make predictions.

Figure 4. Screenshot of Training 2

```
Temperature: 0.2
#####
What do o at at ans o the the at an the the the the at the ing a
e an an the the an the an the the an the an the the at at o the the
the t

What do a the an an an at he an the the the at at at on the an the a
he an the at the the the an be an the inn an an the an the at the an
he at

What do he at at an the an the the at at he anthe inthe athe at the
the at the the the athe as athe the at the the the an the the an th
the
```

After the 20th epoch the model has a much larger vocabulary and is attempting to mimic the structure of a joke.

Figure 5. Screenshot of Training 3

```
40/40 [=====] - 84s 2s/step - loss: 1.8855
Epoch 16/20
40/40 [=====] - 54s 1s/step - loss: 1.8498
Epoch 17/20
40/40 [=====] - 54s 1s/step - loss: 1.8203
Epoch 18/20
40/40 [=====] - 54s 1s/step - loss: 1.8015
Epoch 19/20
40/40 [=====] - 54s 1s/step - loss: 1.7786
Epoch 20/20
39/40 [=====>.] - ETA: 1s - loss: 1.7607####
```

Figure 6. Screenshot of Training 4

```
#####
Temperature: 0.5
#####
What do you call a vere wis the thead in the the cust? Because a he sill does pe.

What do you call a wake for sixper fore? It sand a bord frome.

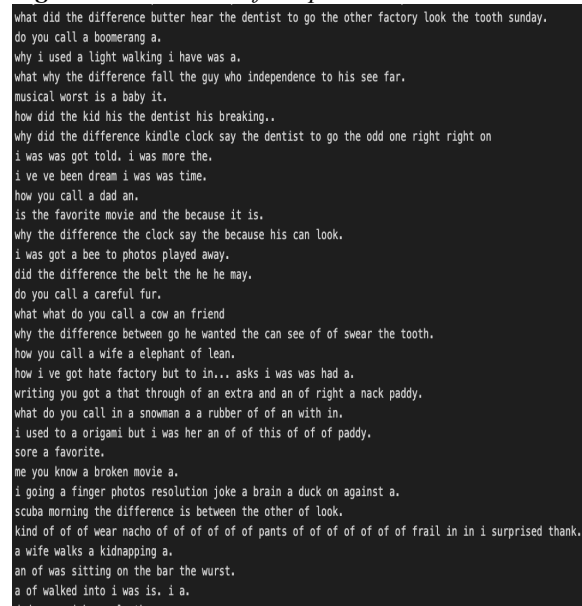
I mas a relay a bay has of the breary? Now the the bat he firtng was to the and a p
I bad of sitt in im and went.
```

Output

The last fully functional version of our program uses our model to generate a text file consisting of 1000 new jokes from our neural network model. The results we got were surprisingly close to an actually good joke. Some of them might elicit a laugh, but only because of how nonsensical they can be. Nevertheless, they lack the requisite coherency and

humor to meet any fair standard of being considered an actual joke.

Figure 7. Screenshot of Output.txt



```
what did the difference butter hear the dentist to go the other factory look the tooth sunday.  
do you call a boomerang a.  
why i used a light walking i have was a.  
what why the difference fall the guy who independence to his see far.  
musical worst is a baby it.  
how did the kid his the dentist his breaking..  
why did the difference kindle clock say the dentist to go the odd one right right on  
i was was got told. i was more the.  
i ve ve been dream i was was time.  
how you call a dad an.  
is the favorite movie and the because it is.  
why the difference the clock say the because his can look.  
i was got a bee to photos played away.  
did the difference the belt the he he may.  
do you call a careful fur.  
what what do you call a cow an friend  
why the difference between go he wanted the can see of of swear the tooth.  
how you call a wife a elephant of lean.  
how i ve got hate factory but to in... asks i was was had a.  
writing you got a that through of an extra and an of right a nack paddy.  
what do you call in a snowman a a rubber of of an with in.  
i used to a origami but i was her an of of this of of of paddy.  
sore a favorite.  
me you know a broken movie a.  
i going a finger photos resolution joke a brain a duck on against a.  
scuba morning the difference is between the other of look.  
kind of of wear nacho of of of of of pants of of of of of of frail in in i surprised thank.  
a wife walks a kidnapping a.  
an of was sitting on the bar the wurst.  
a of walked into i was is. i a.
```

GAN

A Generative Adversarial Network or a GAN consists of two neural network models pitted against each other as adversaries: one, the generator, simulating valid output (the telling of a joke, in our case) and the other, the discriminator, spotting problems with the simulation (i.e., sentences that are not funny). Each neural net takes each output from their adversarial counterpart, and improves subsequent predictions on that basis until any differences can no longer be identified. For our project GAN could help make our model smarter as compared to less complex techniques.

Then the model should be fine-tuned, via transfer learning, to defeat real-text jumbles that are often the result of using relatively small datasets (like ours which consisted only of about 180,000 data points) instead of a large corpus consisting of millions of jokes. Since jokes are subjective, the GAN could provide a window of supervision wherein our generative model learns to better discriminate between jokes that are or are not funny.

GANs are most commonly used for generating hyper-realistic images but have only seen limited use in text generation. The reason is because GANs are designed to produce differentiable values, which makes discrete text generation difficult. Representing text as a value in such a way presents a new problem with its own set of complexities and difficulties. Generative models (creating output data to be evaluated) are considered much more difficult to effectively

implement as compared with the discriminative models (processing/analyzing data for evaluation and classification), though our experience was that getting the two neural nets to communicate meaningfully was our major obstacle in successfully implementing our solution.

References

- [1] Badenhorst, A.E. (2017, December 28). 5 Motivating Reasons Why To Use Chatbots. *BuZZrobot*. Retrieved from <https://bussrobot.com/5-motivating-reasons-why-to-use-chatbots-6610a0b56479>
- [2] McGraw, Peter. Warner, Joel. Do animals have a sense of humour? *Slate*. Retrieved from <https://www.newscientist.com/article/dn25312-do-animals-have-a-sense-of-humour/>
- [3] Harmon, Leon. Teaching AI to Be Funny: The Robot Uprising Won't Be a Laughing Matter. *The Observer*. Retrieved from <https://observer.com/2019/07/teaching-artificial-intelligence-humor-robot-comedy/>
- [4] Waller, Annalu. Black, Rolf. O'Mara, David. Pain, Helen. Ritchie, Graeme. Manurung, Ruli. Evaluating the STANDUP Pun Generating Software with Children with Cerebral Palsy. *ACM Transactions on Accessible Computing*. Retrieved from <https://dl.acm.org/doi/pdf/10.1145/1497302.1497306>
- [5] Hassabis, Demis. Kumaran, Dharshan. Summerfield, Christopher. Botvinick, Matthew. Neuroscience-Inspired Artificial Intelligence. *Neuron*. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0896627317305093>
- [6] Stanford Encyclopedia of Philosophy. Retrieved from <https://plato.stanford.edu/entries/humor/#IncThe>
- [7] Chintapalli, Karthik. Generative Adversarial Networks for Text Generation - Part . Retrieved from <https://becominghuman.ai/generative-adversarial-networks-for-text-generation-part-1-2b886c8cab10>
- [8] Zhang, Yizhe. Gan, Zhe. Fan, Kai. Chen, Zhi. Hénao, Ricardo. Shen, Dinghan. Carin, Lawrence. textGan (Adversarial Feature Matching for Text Generation). Retrieved from