DeepHumor - Generation, Classification and Analysis of Humor

Emil Joswin
112504551
ejoswin
@cs.stonybrook.edu

Aditya Choudhary
112688862
adichoudhary
@cs.stonybrook.edu

Raghav Garg
112907435
raggarg
@cs.stonybrook.edu

Abstract

Humor generation and classification is one the hardest problem in the area of computational Natural Language Understanding. Even humans fail at being funny and recognizing humor. In this project, we attempt to create a joke generator using a large pre-trained language model (GPT2). Further, we create jokes classifier by fine-tuning pre-trained (BERT) to classify the generated jokes and attempt to understand what distinguish joke sentence(s) from non joke sentence(s). Qualitative analysis reveals that the classifier model has specific internal attention patterns while classifying joke sentences which is absent when classifying normal sentences.

1 Introduction

Humor is abstract, high-level use of language that is largely subjective. Even humans fail at being funny and recognizing humor. Humor detection and generation continue to be challenging AI problems. In this paper we aim to explore the possibilities of humor generation and classification through large pre-trained language models. Further, we try to analyse what patterns emerge when a humorous sentence is being classified by the models and use such a model to understand how a joke is different from a normal sentence.

While there have been some advances in automatic humor recognition (Khodak et al., 2017), (Davidov et al., 2010), (Barbieri and Saggion, 2014), (Reyes et al., 2012) , computerized humor generation has seen less progress (Binsted et al., 1997), (Petrovic and Matthews, 2013). Humor generation techniques include generating humor of specific kind like *I like my X like I like my Y, Z* jokes, where X, Y, and Z are variables to be filled in. (Stock and Strapparava, 2003) approached the problem through funny acronym production for given sentences. Also, the current baseline models are based on CNN and RNN with attention and does not utilize recent advances like transfer learning (Howard and Ruder, 2018), or even large pre-trained models like BERT (Devlin et al., 2018) or XLNet (Yang et al., 2019).

To the best of our knowledge there hasn't been any work done in creating generic humor that does not stick to any specific humor stereotype or strict semantics. We try to capture this limitation by relying strictly on unsupervised humor generation. Also, there seems to be a serious lack in studying humor via computational models. In this paper, we try to address both. Mainly we verify the claim that a humor has a punchline in the end that has a strong connection with what is mentioned at the start of the sentence which elicit surprise to the reader thereby making it funny (Attardo, 2010) . We study the attention mechanism of the transformers to verify this claim.

Recently, unsupervised pre-trained language models (GPT-2) have shown great promise in language generation tasks (Radford et al., 2019). Our approach relies on fine tuning large unsupervised language modes like GPT-2 on humor data. We developed our humor classifier based on BERT (Devlin et al., 2018) by fine-tuning over a large corpus of jokes and non-jokes. We believe that they would be successful due to their effective pre-training as well as their due to their ability to learn contextualized representations through transformers. (Rewrite this sentence).

We evaluated our classifier against existing baselines for humor (Rohan Bais) and later evaluated our GPT2 generated jokes over it and compared with non-fine tuned GPT2 that generates generic sentences. Our humor classifier

outperforms the state of the art joke classifier and therefore is a reliable metric for the quality of the joke generated by GPT2. We later analysed the attention patterns in the final layers of the classifier that are specific to joke sentences to study the joke semantics.

Our generative model was fine-tuned over the short-jokes dataset. Due to the subjective nature of jokes, we decided to go with jokes comprising two sentences or less as they had better probability of being funny. Each input instance to the model is one single joke followed by $| < endoftext > |$ tag.

Through this project we show that:

1. Unsupervised humor generation is still a challenge but the jokes generated from newer Language Models like GPT2 significantly improves the quality of joke generation when compared with existing systems.
2. We developed a state of the art humor classifier based on BERT that was later used to evaluate the jokes generated by GPT2.
3. We analysed the attention pattern in our joke classifier and found visible patterns for sentences that are jokes compared with sentences that aren't jokes with remarkable consistency. Later we use this to validate the joke hypothesis that every joke should have a surprise inducing elements that contradicts something that was mentioned at the beginning of the same sentence(s).

2 Our Task

We implement a state of the art joke classifier by fine-tuning pre-trained BERT model (Wolf et al., 2019) with custom generated data for non-jokes. We then, implement a joke generator by fine-tuning GPT2 that is evaluated against this classifier. Challenges for joke generation include controlling the generation via different sampling techniques and imposing penalties on bad token generation. Unlike the previous joke generation methods, our aim is to create a generic joke that does not follow a particular joke pattern.

2.1 Joke Generation

We use GPT2 as our pre-trained model for joke generation. Below we give a high level overview of the fine-tuning performed over GPT2. The details are described in section 3.1.

2.1.1 GPT-2

The fine-tuning task on GPT2 is setup as a problem of language model fine-tuning with the short-jokes dataset. We try to predict the next word of sentence in an auto-regressive manner. From the dataset, we remove jokes that are more than two sentences in length. This was done in order to effectively evaluate our classifier since most of the joke - no joke data that we have follow this pattern. We control the joke generation using sampling methods like top-k (Fan et al., 2018) and nucleus sampling (top-p) (Holtzman et al., 2019) . We also penalize repeated generation of the same tokens (Keskar et al., 2019).

2.2 Joke Classification

For our joke classification task, we propose two baseline models. In this section we give a brief description of these models and the issues that we are trying to improve.

2.2.1 Baseline Model (s)

First, we utilized a CNN model which shows success in several text categorization tasks (Johnson and Zhang, 2015). The model represents sentences as stacked word embeddings. It will retrieve a matrix of word embeddings representing a sentence before proceeding to perform convolutions on this matrix. We will use convolutions on regions of 2, 3, 4, and 5 words. The idea here is to capture adjacent words in a "window" and then retrieve a vector that represents the word and it's neighbors via convolutions. We then perform a max-pool on the outputs and concatenate the poolings into a vector, and use this to feed into a fully connected layer that outputs the probability of the 2 classes [Figure 1]

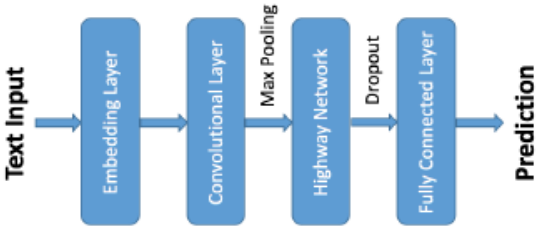


Figure 1: CNN Network Architecture

Second, we utilized a RNN model with attention. We feed the sentence into an embedding

S.No.	Prompt	Type	Output	Score
1.	When did Obama become	Joke	When did Obama become the famous black guy? When he was in prison.	0.999
		Non-Joke	When did obama become president of the usa?	0.356
2.	I love NLP	Joke	I love NLP. It is the only class I can do.	0.985
		Non-Joke	I love NLP even though I don't know what it is.	0.568
3.	When is	Joke	When is a door not a door? When it's ajar.	0.985
		Non-Joke	When is the newmoon?	0.312
5.	What do you call a	Joke	What do you call a Dinosaur with Diahrrea? A shity-sore-ass	0.998
		Non-Joke	What do you call a person with no name?	0.671
6.	Joe stood up and spoke to the	Joke	Joe stood up and spoke to the man in the wheelchair and said I'm not going to stand up for you.	0.015
		Non-Joke	Joe stood up and spoke to the crowd.	0.008
7.	Obama flew to Mexico	Joke	Obama flew to Mexico and asked for a Juan	0.612
		Non-Joke	Obama flew to Mexico.	0.471
8.	Isn't language learning	Joke	Isn't language learning like being a prostitute? It's a lot of work for a prostitute.	0.999
		Non-Joke	Isn't language learning is a good thing. It's a good thing.	0.879
9.	When did Aditi	Joke	When did Aditi go to college? At the end of the day.	0.872
		Non-Joke	When did Aditi tell you?	0.448
10.	Do you know what happened to	Joke	Do you know what happened to the guy who got his left side cut off? He's all right now.	0.998
		Non-Joke	Do you know what happened to the other guys?	0.034

Table 1: GPT2 generated Jokes and Non-Jokes along with their Classification scores.

layer and pass each of the embeddings as inputs to the RNN. For each timestep, we are then returned a hidden state. For each of these hidden states we want to calculate attention to see which of the words are more important, so we feed each hidden state into a multiplicative attention mechanism. The RNN is similar to what was proposed to Oliviera's model (de Oliveira and Rodrigo, 2015), only attention mechanism was added in order to improve the vanilla RNN data fitting.

2.2.2 Drawbacks of Baseline (s)

Our baselines model captured different features of the sentence structure individually. CNN was able to capture word window features while RNN with attention has the ability to capture time-based data retention. But they were not complex enough to capture both the features at the same time. Also, they have inherent limitations when it comes to language model pre-training and it has been shown that transformer based models outperform most other models in almost every NLP task due to their advanced context representation capabilities.

3 Approach

Generative pretraining of sentence encoders are shown to work well in NLP tasks like sentence generation and classification. In our experiments we use pretrained transformer models from Huggingface(Wolf et al., 2019) for fine-tuning them for joke generation and classification respectively.

3.1 Fine-tuning and joke generation using GPT2:

We use distilGPT2 as our pretrained transformer model due its smaller parameter size. GPT2 is based on the original GPT2 but has reduced parameter size due to knowledge distillation. GPT2 is a large unsupervised Transformer language model. It was trained on the WebText corpus which contains slightly over 8 million documents with a total of 40 GB of text obtained from URLs iin Reddit submissions with at least 3 upvotes. GPT2 uses byte-pair sentence encodings. The sentence embedding has a dimension of 768. The model consists of 6 layers with 12 attention heads each. We limit the sequence length to 30 for our task. Any sentence that is longer than that is ignored. If a sentence is smaller than 30 tokens long, we pad $| < endoftext > |$ at the end. The generation is controlled via top-k and top-p sampling. We use a top-k value of 50 and pick only the top 50 logits from the model output. Out of these 50 tokens we pick the hightest tokens which sum to a total top-p value of 0.8. A repetition penalty (Keskar et al., 2019) of $1/1.2$ is multiplied with the logits in order to penalize repeated generation of same tokens again and again. The results are provided in Table 1.

3.2 Implementing the joke classifier

We have selected BERT's *base-uncased* as our pre-trained model. BERT is a multi-layer bidirectional Transformer encoder and was initially trained on a 3.3 billion word corpus. We then fine-tune it with the joke-nojoke training set to output probability for the two classes. We have limited to the sequence length 42, and have trimmed the sentences which were longer. We chose to use this transformer based model because of its success at recognizing and attending to the most important words in both sentence and paragraph structures. We have trained our model for three epochs.

3.3 Analysis of attention values of the classifier

A good way to understand the transformer is by visualizing the attention patterns for individual attention heads in the model. The attention mapping of the sentences generated by the classifier could be visualized on the basis of the magnitude of the attention values (see Figure 2). It would give a good indication which word pairs have the strongest relation with each other while capturing the sentence semantics as a whole. The attention values of the classifier is visualized using (Vig, 2019). It was found that there is a visible pattern in the shape of 'X' towards the last 2 layers of the classifier which is absent for non-joke sentences. This indicates that the words of phrases occurring at the end of the sentence has got a strong connection with what was mentioned at the beginning of the sentence. This validates the claim that a joke generally consists of a "setup" followed by a sudden surprise element at the end relating the start of the sentence (Attardo, 2010). This pattern is observed with strong regularity for jokes and hints that for a sentence(s) to be a joke there should be word or a phrase at the beginning of a sentence that the end is referring to. This pattern isn't observed for non-joke sentences. See Figure 4.

3.4 Implementation Details

Our overall model architecture looks as described in the Figure 3. A prompt is fed to the fine tuned GPT2 model which completes the sentence by trying to make it funny. This completed sentence is then fed to the joke classifier which classifies it as either joke or non-joke. We also analyse the attention of the each individual head of the last two layers of the classifier.

Figure 2: Sample Attention pattern for one single head. The thickness of the lines indicates how strong the attention is in between two words of the same sentence.

4 Evaluation

Our classifier is evaluated against the baseline models and was found to outperform them to obtain an accuracy of 0.983. Later out GPT2 generated sentences were evaluated using this classifier. During classification task, we assume that the sentences generated using our fine-tuned model is a joke, whereas the ones generated by the pre-trained model is a generic non-joke sentence. The classifier gives an accuracy of 0.74 which is a surprisingly good score for an unsupervised generative model. The attention analysis for a given sentence is based on the attention heads of the classifier while it tries to classify a sentence as joke or no-joke.

4.1 Dataset Details

To fairly evaluate the performance on humor recognition, we need the dataset to consist of both joke (positive) and non-joke (negative) samples. For the positive samples, we used the Short Jokes (Moudgil) dataset and for the negative samples, we choose WMT162 English news crawl as our non-joke data resource.

4.1.1 Short Jokes Dataset

The short jokes dataset have been taken from an open database on a Kaggle project (Moudgil). It contains 231,657 short jokes with no restriction on

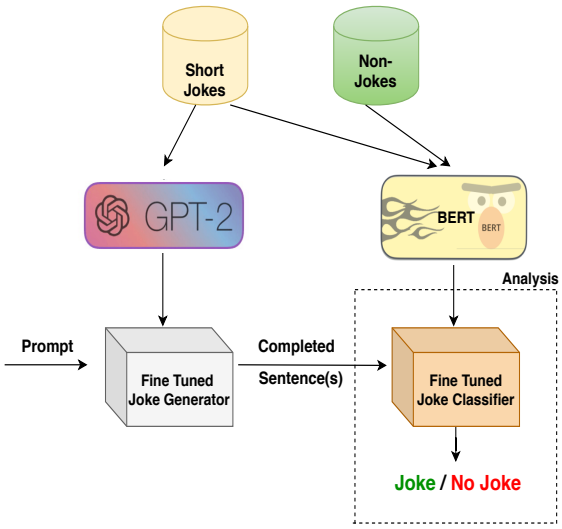


Figure 3: Model Architecture

joke types scraped from various joke websites and length ranging from 10 to 200 characters.

4.1.2 WMT162 English News Crawl

We choose WMT162 English news crawl as our non-humorous data, similarly utilized by (Chen and Soo, 2018). However, simply treating sentences from the resource as negative samples could result in deceptively high performance of classification due to the domain differences between positive and negative data. So we are trying to pick examples whose words all appear in the positive samples and whose word count equals the length of the average text length of the joke sentence. This was done in order to match the two halves (jokes and non-jokes) as closely as possible.

4.2 Evaluation Measures

We use accuracy to measure the effectiveness of the classifier as our data is balanced.

4.3 Baselines

We ran the CNN model with windows of up to 5 words, and used 1-dimensional convolutions with max-pooling. The results are concatenated before feeding into a fully-connected layer. Both CNN and attention-based RNN models were run for 50 epochs. By trying out different batch sizes, learning rates and dropout rates we found that the most ideal hyper-parameters were learning rate of 0.1, batch size of 32, dropout of 0.3. CNN achieves a baseline accuracy of 0.906 where as RNN + attention achieves 0.921.

Method	Accuracy	Precision	Recall	F1
CNN	0.906	0.902	0.946	0.924
RNN + attention	0.921	0.918	0.935	0.926
BERT	**0.983**	**0.983**	**0.983**	**0.983**

Table 2: Table comparing our joke classifier with the baseline models. Our model outperform both the CNN and RNN+attention models

4.4 Results

Our best model based on BERT is able to achieve an accuracy of 0.983 which is a significant improvement over the baseline. The classification results of the joke classifier are reported in Table 2. The results of joke generation and classification are summarized in the Table 1.

4.5 Analysis

1. We were able to outperform the current baseline for joke classification by large margin which points out the effectiveness of pre-trained language models based on transformers.

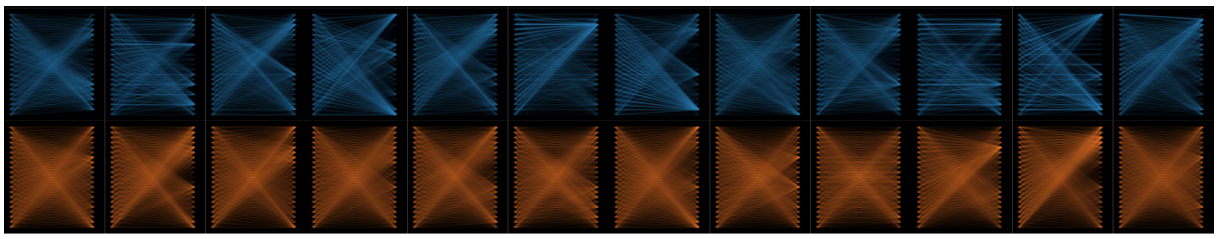
2. Occassionally, GPT2 generates sentences with a question mark in the middle which tricks BERT into classifying it as a joke. This problem could possibly be eliminated by adversarial training.

3. We were also able to show that our classifier is able to capture 'X' shape attention pattern for jokes sentences which validates the hypothesis that jokes generally follow a "setup" and "punchline" structure wherein there would be strong associations between one or two words at the beginning and the end of a sentence. This association captures the shock/surprise which results in humor.

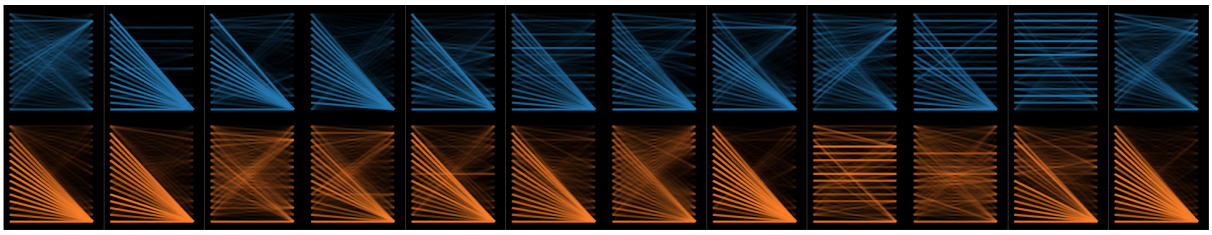
4. We also tried to see if the length of the prompt given to the GPT2 makes any difference in the effectiveness of a joke. From Figure 5, we find that the maximum accuracy is found for prompts of length 3 and 8 and minimum is found for prompts for length 9.

4.6 Code

Our code, dataset and instructions to reproduce the results are mentioned at `https://github.com/adich23/Deep-Humor`



(a) Attention pattern for joke sentence(s)



(b) Attention pattern for non-joke sentence(s)

Figure 4: Visualizing attention patterns of the last two layers of the joke classifier. Each layer has 12 heads. It can be clearly seen that for joke sentences, their attentions when visualized looks like 'X' shape. This clearly hints that the end of the sentence has a strong connection/association with the beginning of the sentence. This particular pattern is not observed in normal sentences.

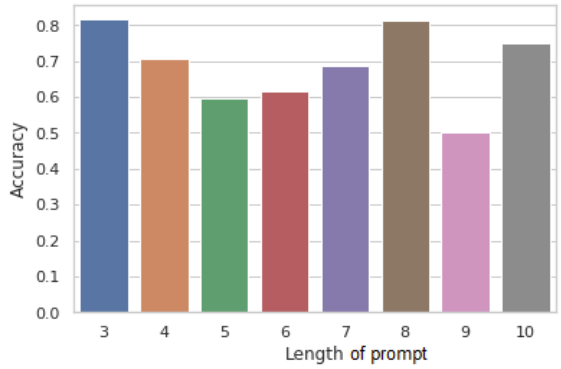


Figure 5: Prompt length vs Accuracy of being classified as a joke

5 Conclusion

In this paper we were able to create a new joke classifier that beat the previous existing baseline using large pre-trained models. We also proved that it is possible to create a high quality joke generator without relying on hard-coding joke sentence semantics into the model by using large pre-trained language models like GPT2. We were also able to capture the hypothesis that most jokes have a setup and punchline structure when compared with normal sentences to it by analysing the attention patterns of the last layers of the classifier.

In our experiments we found that GPT2 was able to trick BERT into classifying a sentence as a joke if it has a question mark somewhere in the second half of the sentence. We believe a better joke generator could be created by employing adversarial training techniques where the joke generator competes agains joke classifier.

References

Salvatore Attardo. 2010. *Linguistic theories of humor*, volume 1. Walter de Gruyter.

Francesco Barbieri and Horacio Saggion. 2014. Modelling irony in twitter. pages 56–64.

Kim Binsted, Helen Pain, and Graeme Ritchie. 1997. Children's evaluation of computer–generated punning riddles. *Pragmatics & Cognition*, 5.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 107–116, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Angela Fan, Mike Lewis, and Yann N. Dauphin. 2018. Hierarchical neural story generation. *CoRR*, abs/1805.04833.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *CoRR*, abs/1704.05579.

Abhinav Moudgil. Short jokes dataset, kaggle.

Luke de Oliveira and Alfredo Lainez Rodrigo. 2015. Humor detection in yelp reviews.

Sasa Petrovic and David Matthews. 2013. Unsupervised joke generation from big data. In *ACL*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Chin Reyes, Marc Brackett, Susan Rivers, Mark White, and Peter Salovey. 2012. Classroom emotional climate, student engagement, and academic achievement. *Journal of Educational Psychology*, 104:700–712.

Marcos Torres Rohan Bais. Deep learning humor classification on yelp reviews.

Oliviero Stock and Carlo Strapparava. 2003. Getting serious about the development of computational humor. pages 59–64.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding.