# Transformer based Neural Joke Generator

Taaha Kazi
Don Bosco Institute of Technology
Mumbai
Maharashtra, India

Sameer Joshi
Don Bosco Institute of Technology
Mumbai
Maharashtra, India

Steeve Kaitharath
Don Bosco Institute of Technology
Mumbai,
Maharashtra, India

Imran Ali Mirza
Don Bosco Institute of Technology
Mumbai,
Maharashtra, India

## ABSTRACT
Humor is a complex and intrinsic part of human conversation, which involves a deep understanding of grammatical structure and knowledge of the world. Building computational models that can identify and generate humor remains a challenging field. This work presents a neural network based joke generator that employs a transformer-based architecture. To improve the generator's performance, the model was further trained with Proximal Policy Optimization (PPO), a reinforcement learning algorithm. The model's performance was evaluated by human ratings by conductingqualitative analysis.

## General Terms
Natural Language Processing, Reinforcement Learning, Transformers

## Keywords
Natural Language Generation, Humor

## 1. INTRODUCTION
Humor is one of the most evolved characteristics we possess. While it is a complex linguistic tool, it is instinctive and straightforward to understand and create. The exact reasons why we laugh and what makes us laugh are still relatively unknown. The ability to understand and write jokes requires an understanding of different concepts. One must be aware of the world and the relationship between ideas and objects and be aware of linguistic structures that make a sentence funny, i.e., a strong understanding of semantics and syntax is required to understand and generate a joke.

Even though understanding humor or any aspect of language is complex, advances in natural language processing and neural network architecture have led to success in tasks such as text classification, generation and summarization. Also, pre-trained language models such as BERT[1]and GPT[2] trained on a large corpus have shown comprehensible knowledge about the world.

This paper proposes a method to leverage the capabilities of pre-trained language models to build a joke generator. The proposed approach is first to fine-tune a pre-trained generative model on a dataset of jokes. This is followed by building a joke identifier that can identify whether a given sentence is a joke or not. Then, additionally train the generator with the identifier to improve the quality of the joke. This additional training of the generator with the identifier is done with Proximal Policy Optimization,[3] a reinforcement learning algorithm.

## 2. RELATED WORK
This work focuses on building a joke generator, which involves building a joke identifier, a joke generator, and a training mechanism to train them together. Some of the existing literature in humor identification is presented, followed by work in text generation and methods to train a generator.

For humor recognition, experiments have been carried out with different methods. Yang et al. [4] experimented with Word2Vec combined with K-NN Human Centric Features for humor recognition. Weller and Sippi [5] proposed a transformer-based architecture for identifying jokes. Annamoradnejad and Zoghi [6] employed BERT sentence embedding for humor detection. They used BERT to generate embeddings for sentences of a given text and input these embeddings to a neural network that processes sentences separately in parallel hidden layers.

Ziegler et al. [7] fine-tuned pre-trained language models with reinforcement learning, which used a reward model trained from human preferences to improve performance. They used this method for text completion tasks, which led to an improvement in performance.

## 3. METHODOLOGY
The joke generator was built in three steps. First, the generator model is trained on a dataset of jokes. Then the joke identifier is trained to identify if a given sentence is a joke or not. Finally, the generator is trained with the joke identifier to improve the quality of the jokes generated. This process is depicted in figure 1.

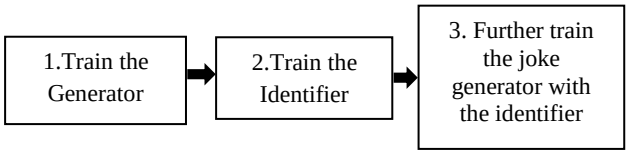


**Fig 1: Training flow**

### 3.1 Joke Generator
For conducting experiments, GPT-2 small was used as the base model. GPT-2 model can be fine-tuned with a dataset to mimic the style and logic of the given dataset. GPT-2 is anauto-regressive generative language model built using transformer decoder blocks trained on a dataset of 8 million

web pages and imbibes a deep understanding of the world. It also displays excellent capability in both long and short-form text generation. The model's architecture is based on Vaswani et al. [8]transformer model, shown in figure 2. The transformer consists of an encoder stack and a decoder stack. The input to a transformer first goes through a multi-headed self-attention layer followed by a feed-forward network. The output from the encoder is passed to the decoder, which consists of a similar self-attention mechanism and feed-forward network, along with a masked attention layer.

The model was fine-tuned on the joke dataset provided by Weller and Sippi [5]. 173,635 sentences from the dataset labeled as jokes were used for fine-tuning. This dataset included jokes with multiple sentences from the Short Jokes dataset[9] and single sentences from Pun of the Day[4] dataset. Preprocessing on the dataset was carried out, which included removing sentences shorter than 10 characters and dropping the duplicate ones. The sentences were then tokenized using the GPT-2 tokenizer. The model was trained with a learning rate of 5e-4 for 5 epochs.

## 3.2 Joke Identifier

For the joke identifier, BERT-base model was used, a causal language model is built using transformer encoder blocks. BERT base consists of 12 Encoders with 12 bidirectional self-attention heads and is trained on a dataset extracted from the. It excels in tasks related to natural language understanding. When it was released, it achieved state-of-the-art results in tasks such as GLUE (General Language Understanding Evaluation) and SQuAD (Stanford Question Answering Dataset). The model's architecture is based on Vaswani et al. [8]Transformer model but only consists of the encoder blocks.

The model was fine-tuned on the downstream task of classifying a given sentence as either a joke or not a joke. The same dataset that was used for training the generator is used to train the identifier to avoid domain mismatch. A balanced dataset of 173,635 jokes and 173,851 non-joke sentences was used. This dataset included jokes with multiple sentences from the Short Jokes dataset[9] and single sentences from Pun of the Day [4] dataset. The dataset was then split with 80 percent reserved for training and the remaining 20 percent for validation. For preprocessing, sentences shorter than 10 characters were removed, and duplicate sentences were dropped. Punctuation marks were removed as they were easy lexical indicators of jokes, such as a question mark at the end of knock-knock jokes. The sentences were then tokenized using the BERT tokenizer. The model was trained with a learning rate of 2e-5 for 2 epochs.

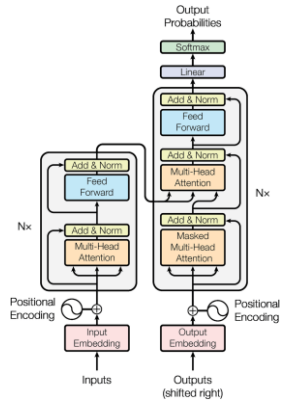


**Fig 2: Transformer Architecture**

## 3.3 Proximal Policy Optimization

To improve the quality of the generator, the output of the joke identifier was incorporated while training. Ziegler et al.[7] applied Proximal Policy Optimization for fine-tuning pre-trained language models. This paper adopts a similar methodology to improve the generator by further fine-tuning it with the joke identifier.Before optimizing the generator, we create a replica of the generator, which will be helpful during optimization. This step is carried out in three steps

### *3.3.1 Generation*

The joke generator will generate a joke.

### *3.3.2 Joke Evaluation*

The joke generated is evaluated by the joke identifier, and it generates a score denoted by '$\boldsymbol{r}$'.

### *3.3.3 Optimization*

It is further divided into three steps, shown in figure 3.

#### *3.3.3.1 Calculating KL Divergence*

Log probabilities of the generated joke are calculated with the original generator and the replica. The KL-divergence between the log probabilities is calculated and denoted as '$\boldsymbol{KL}$'.

#### *3.3.3.2 Updating Reward Score*

The updated reward score R considers the KL Divergence score, i.e., $\boldsymbol{R = r - \beta KL}$, where $\boldsymbol{\beta}$ is the KL divergence coefficient.

#### *3.3.3.3 Model Update*

The generator is trained via PPO with an updated reward $\boldsymbol{R}$.

The KL divergence is used for optimization as it measures the difference in the log probability distribution, which ensures the generator does not deviate far from its original form. A similar approach by Kazi [10] for detoxifying language models

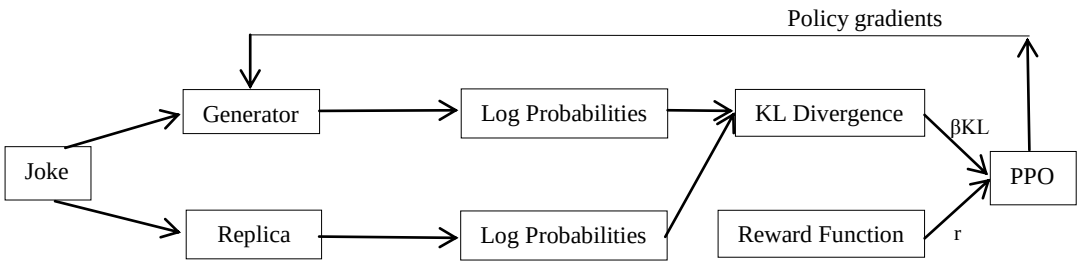


**Fig 3: Optimization Step**

We further trained the model in the proximal policy setting for 1 epoch with KL coefficientβ=0.2 and a batch size of 256.

# 4. RESULTS

The qualitative and quantitative impact of fine-tuning the generator with Proximal Policy Optimization were carried out. The analysis of the results show that the generator model was able to understand the fundamental concepts and phrases of humor and was able to replicate human language patterns to some degree. Although it wasn't coherent all the time, it was able to generate humorous fragments.

## 4.1 Quantitative Results

For measuring the quantitative impact, the ratio of the score the joke identifier assigns to the model before the optimization and after the optimization was calculated. Plot of the graph over the training steps is shown in figure 4.

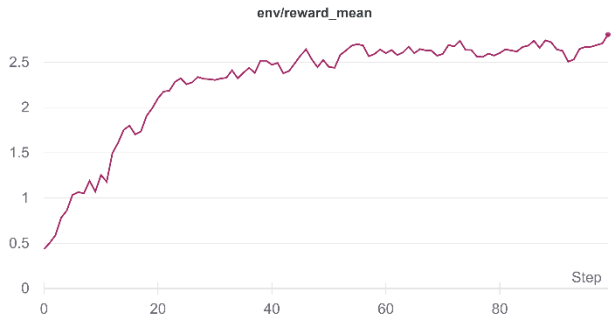


**Fig 4: Reward ratio over training steps**

The graph shows a clear increase in the value of the ratio, which indicates that Proximal Policy Optimization leads to measurable improvement in the quality of the jokes generated.

For testing, the identifier classified a random set of 200 sentences generated by the model. The identifier classified 145 sentences as funny and the remaining as non-funny.

## 4.2 Qualitative Results

Like other natural language generation models, the model did not always perform as intended. It occasionally suffered from overfitting, i.e., it would repeat the same joke that it had been trained and barely changed the words in the joke. It also would generate gibberish sentences from time to time. Apart from this, the model was able to learn the styles of jokes it was trained on and generated some hilarious jokes.

Some examples of the above-mentioned observations:

Example of the model overfitting and almost copying a joke:

*Joke Generator: I used to be addicted to soap... i am clean now*

Example of the model generating normal sentences, i.e. something that was not funny:

*Joke Generator: I have kids and i have to go home*

Example of the model generating a gibberish sentence:

*Joke Generator: the first rule of fight club is do not talk to each other. i heard it was a jam*

Example of the model generating a joke:

*Joke Generator: What do you call a communist animal, a moscow.*

Here, the model displays a deep understanding of words and the world. It is able to connect the concepts of communism, the city of moscow, and the name of an animal.

## 4.3 Joke Identifier Accuracy

Different approachs for joke identification were tested. The BERT based joke identifier performed better than all other techniques and achieved a competitive accuracy. The accuracy scores for all the approaches are mentioned in the table 1.

**Table 1. Comparison of different methods for joke identification**

| Approach | Accuracy |
|---|---|
| Decision Tree | 77.84 |
| CNN | 85.44 |
| **BERT base** | **95.83** |

## 4.5 Joke Generator Scores

A random set of 100 sentences generated by the model were subject to human evaluation. The evaluators graded the jokes on a 3 point scale: 0 (Incoherent), 1 (Somewhat funny), and 2 (Funny). The sentences generated were judged by 10 human evaluators. The aggreate results of all the evaluators are mentioned in table 2.

**Table 2. Human evaluator's scores for the jokes generated by the model**

| Label | Percentage |
|---|---|
| Incoherent | 27.1 |
| Somewhat funny | 46.2 |
| Funny | 26.7 |

# 5. CONCLUSION AND FUTURE WORK

In this paper, we showed methods to build models that can carry out humor identification and generation. The proposed model for humor identification achieved a competitive score of 95.83%. We explored training the joke identifier and generator together to improve the joke generation scores. This method of training does not require human intervention and leads to improved scores. The generator, to a great extent, was able to generate short funny jokes. This model can be used to assist humans in generating jokes and humorous sentences.

For future work, we can develop this system based on a business use case. We can fine-tune it to a specific domain dataset, such as advertising, editing, and the model can generate funny taglines or creative slogans.

# 6. REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. North American Chapter of the Association for Computational Linguistics.

[2] Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya. 2019. Language Models are Unsupervised Multitask Learners.

[3] John Schulman and FilipWolski and PrafullaDhariwal and Alec Radford and Oleg Klimov 2017. Proximal Policy Optimization Algorithms.

[4] Diyi Yang, AlonLavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,

[5] Orion Weller, Kevin Seppi. Humor Detection: A Transformer Gets the Last Laugh. 2019. Association for Computational Linguistics

[6] IssaAnnamoradnejad. ColBERT: Using {BERT} Sentence Embedding for Humor Detection. 2020. https://arxiv.org/abs/2004.12765

[7] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown ,Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving 2020. Fine-Tuning Language Models from Human Preferences

[8] Ashish Vaswani, Noam Shazeer, NikiParmar, JakobUszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and IlliaPolosukhin. 2017. Attention Is All You Need. 31st Conference on Neural Information Processing

[9] AbhinavMoudgil. Short Jokes. 2016. https://www.kaggle.com/abhinavmoudgil95/short-joke

[10] Taaha Kazi. Detoxifying Language Models with Proximal Policy Optimization. 2021. Manuscript in preparation.