What a Joke (Generator)

Allen Chen, Jerry Chen, Sakshi Madan, Shun Lin

CS 182 | Spring 2019

Introduction

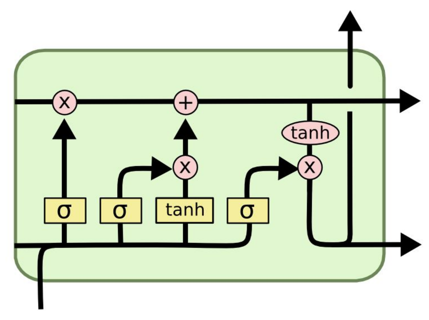
- **Computational humor** is a branch of computational linguistics and artificial intelligence which uses computers in humor research. It is a relatively new area and therefore only a handful of research paper are currently tackle this field[1].
- In our project, we explore different models to generate jokes. We built a **language model,** a **transformer model**, an **encoder-decoder model** and model using **pre-trained GPT**.

Objectives

- The **primary objective** for our project is to build a model that can generate good jokes that will bring the audience laughter and joy.
- Due to the time and capacity constraint we have, we measured each model's performance by evaluating the **training & validation loss per couple thousands of iterations** instead of conducting a survey with our generated jokes to collect score from real people**.**
- We also look over some generated jokes from our models to **evaluate the effectiveness** of each of our model on joke generation task.

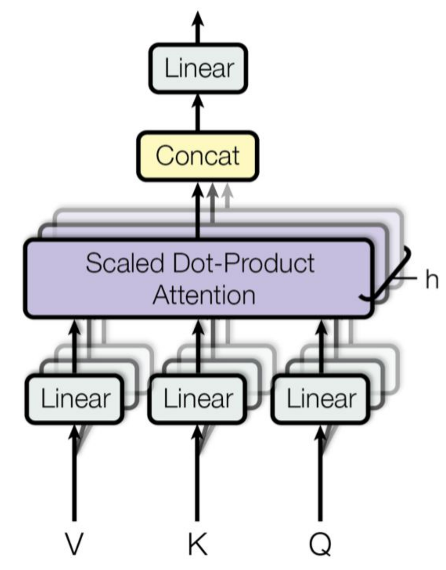
Models

- **Basic LSTM Model**
  - For the basic LSTM model, we concatenated all jokes as input to our model.
  - We then used a single LSTM to generate answers using Keras.



LSTM Cell

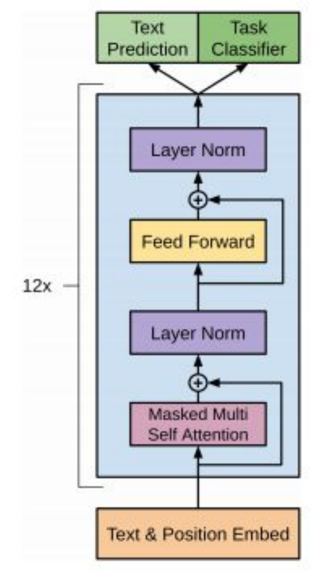
- **Language Model**
  - A language model is a neural network that assigns a probability to each sequence of words.
  - Our basic language model uses a single LSTM to generate the next word of the answer to a joke.



Multi-head Attention

- **Encoder-Decoder**
  - Sequence-to-sequence model which uses two RNNs.
  - During training, encoder input is the embedded question, and the decoder is fed the embedded answer.
  - During inference, we ask it a question to produce an answer.

- **Transformer**
  - Our Transformer network uses multihead attention, allowing each input to attend to at many places at once[2].
  - Similar to the encoder-decoder model, during training, we have questions as input to the encoder, and answers as input to the decoder.



GPT-2 Model

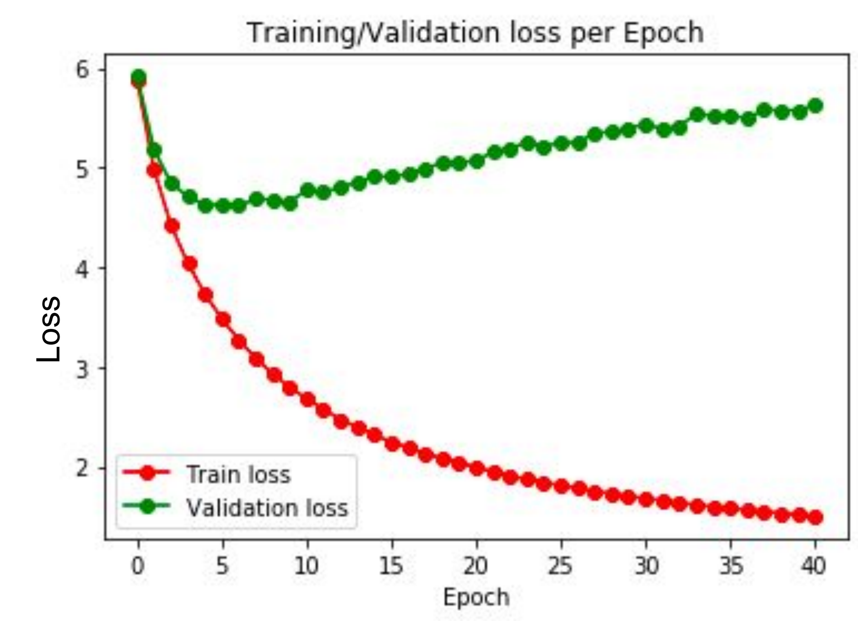
- **Pretrained GPT-2 with Finetuning**
  - We used a pretrained GPT-2 transformer decoder based language model, with OpenAI weights.
  - Then, we fine-tuned this model on our joke dataset.

Data Source

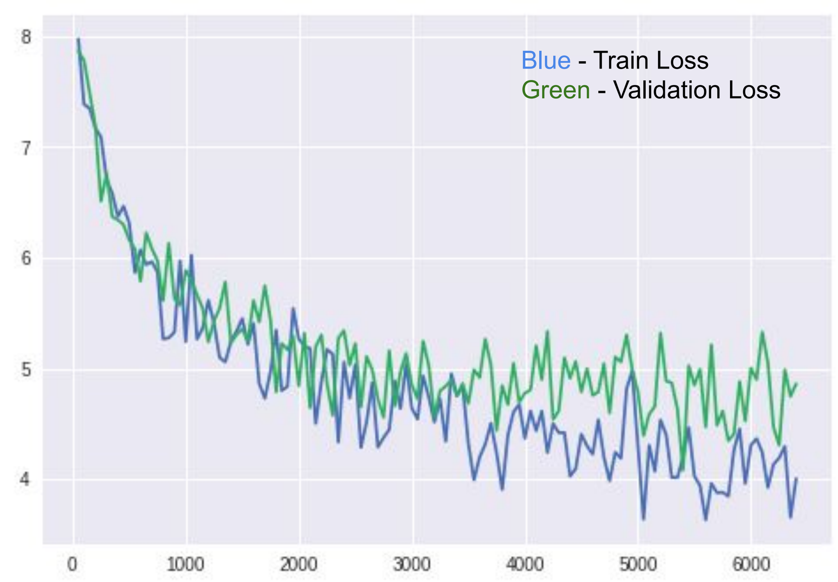
- We gathered our jokes data from multiple sources including **Kaggle**, **Github**, and the **Jester** dataset from **Berkeley Research**.
- We filtered out jokes that contain bad and racist words, as well as jokes that do not make sense or have typos.
- We preprocess our data differently for different models. For example in our earliest **LSTM** model we concate jokes together and train the model over one sequential text. On the other hand, we preprocessed and tokensized our data into expected input to **GPT** for our model that utilize pre-trained GPT model.
- We obtained a pre-trained GPT model from **OpenAI's** github and paper[2].
- We obtained a pre-trained BERT model from Google Research's Github and paper[3].

Results

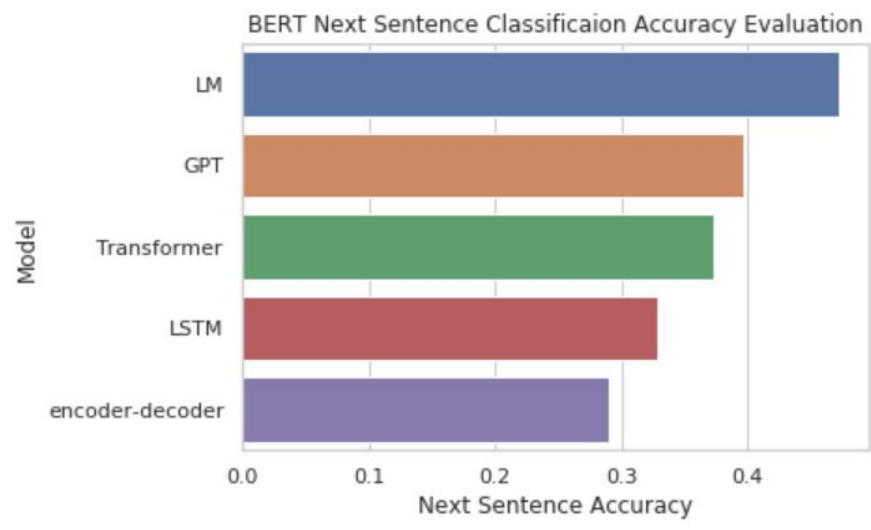
Encoder & Decoder



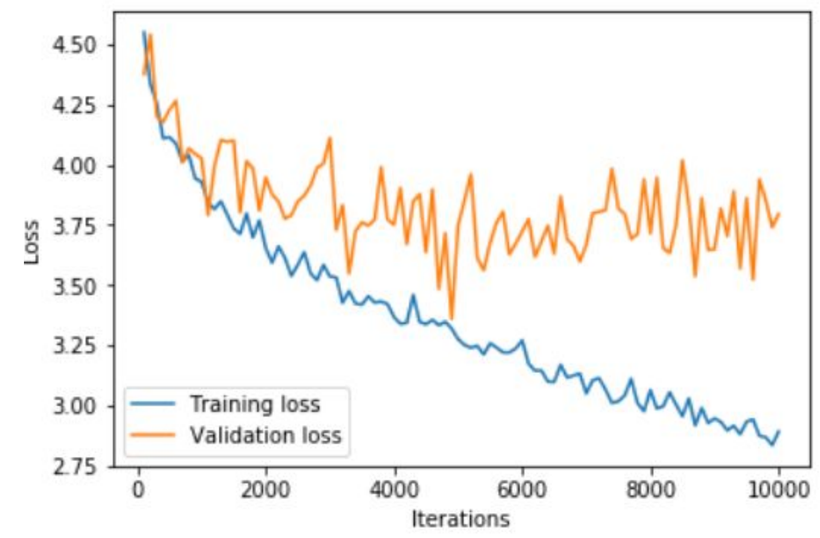
Transformer



BERT



GPT



Sample Jokes

- **GPT:**
  - **Why do chickens have hooves instead of feet?** *Cuz they have hooves instead of feet.*
  - **How do you make a baby cry twice?** *You slap the hell out of it.*
- **Transformer:**
  - **Which came first the chicken or the egg?** *The the rorooosterster did did the the fit*
- **Language Model:**
  - **knock knock who's there** *! ! ! who ? ! who ? i don't know , i just want to be a little bit . . .*
- **Basic LSTM Model:**
  - **Why did the chicken cross the road?** *Because he didn't want to watch TV to someone who had a moment when she was a second language.*
- **Encoder-Decoder:**
  - **When are they going to drug test the audience of the price is right?** *losers policy wallaby sculptures duhh shouldve hooking untill quickie investment guinness*

Conclusions

- By comparing the above graphs, we see that the **GPT-2 Model** has the **lowest validation loss**. Most generated jokes ended up either not being funny, or does not make sense. However, our models do generate a few good jokes.
- One main lesson we learned from completing this project is that it is essential to explore different models and approaches to solve the computational humor problem.
- One of the challenges we faced was **preprocessing the data** and determining proper **hyperparameters** to train a good model.
- Since our dataset was fairly small, it was difficult to create answers to jokes that were meaningful or made sense most of the time. Thus we also used the pre-trained **GPT-2 model** with finetuning. As we can see from the above sampled jokes, jokes generated by our **GPT-2 model** does make more sense than others.

References

[1] Kim Binsted. "Computational Humor". University of Hawaii. March/April 2016. URL:http://www2.hawaii.edu/~binsted/papers/BinstedetalIEEEComputationalHumor2006.pdf

[2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever. "Language Models are Unsupervised Multitask Learners". OpenAI.

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" October, 2018. URL: https://arxiv.org/abs/1810.04805