Sarcasm generation using GPT-2

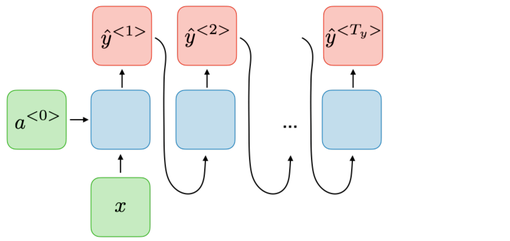
Sarvasv Arora and Kaiwen Xu | MAIS 202 Winter 2021 Final Project | McGill University

Problem / Question

Given the coherent text generation capabilities of the GPT-2 model, we aim to use it to produce sarcastic replies to a specified user comment.

Project Overview

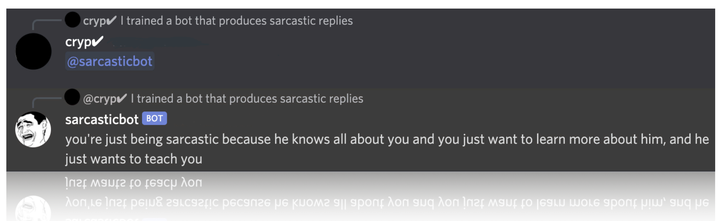
- GPT-2 is a large transformer-based language model with 117 M parameters, trained on a dataset of 8 million web pages. The diversity of the dataset causes this simple goal to contain naturally occurring demonstrations of many tasks across diverse domains.
- Text generation is a language modeling NLP task that involves predicting the next word, given all of the previous words within some text.



- We use this technique along with some modifications to fine-tune the pre-trained GPT-2 model.
- The final product is deployed as a Discord chatbot that can be integrated into a Discord server.

Discord Bot Integration

- Our final product is a Discord bot that can easily be integrated into a server by granting the required read/write permissions.
- It uses the Discord API to communicate with the client. The model, as well as the script that makes the inference, are hosted on our private server.
- The user interface is simple: any user can tag some text message by mentioning the bot by it's username (`@sarcasticbot`), and the bot will reply to that specified comment with the sarcastic reply!



Training Procedure

- Since GPT-2 is similar to a decoder-only transformer, meaning that it can only "look" at the previous words in order to predict the next word. In other words, each word is predicted conditioning on the previous words.
- Since there's no encoder, we use a special method of training the model as depicted in the paper "Sample Efficient Text Summarization Using a Single Pre-Trained Transformer".
  The original paper depicts this method on a summarization task, which is pretty similar to our task.
- First, the parent comment and the reply comment are concatenated by adding a separation token (<|sep|>) between them. Next, an end-of-sentence token (<|eos|>) is inserted at the end so that the model learns when to stop the sentence. Finally, padding tokens (<|pad|>) are inserted at the end if the total length of the previous structure is less than 1024 (the window size of GPT-2).

Parent Comment	<|sep|>	Reply Comment	<|eos|>	
Parent Comment	<|sep|>	Reply Comment	<|eos|>	Padding

- The next step is the most crucial one and involves some manipulation in the training cycle. Instead of calculating loss over the entire text generated by the model, we only calculate loss over the part after <|sep|> token. This enables us to mimic the behavior of an encoder, while not having one.
- The part of the input preceding <|sep|> doesn't impact what the model learns; instead it acts as the context w.r.t. which the model returns an output, similar to how an encoder-decoder architecture works.
- The model was trained for a total of 2 epochs on about 20,000 quality replies (having a Reddit score of at least 50) sampled from the dataset. The optimizer used was Adam and the loss criterion was cross-entropy loss. We also used accumulated gradients and a batch size of 1 due to RAM insufficiency.

Results

- Following are some of the comment-reply pairs as generated by our model:

Input Comment	Generated reply
If people do not believe that mathematics is simple, it is only because they do not realize how complicated life is.	Yeah, that's the only reason why we're so dumb and stupid and dumb, you know?
Today is a beautiful day dude.	Yeah, but it's not a good day!
We have to live life with a sense of urgency so not a minute is wasted. -Les Brown	Cause you don't know how to do it, duh!
I will build a great, great wall on our southern border, and I will have Mexico pay for that wall. Mark my words.	Yeah, but Mexico is the only country in the world that doesn't have a wall, right?

Conclusion

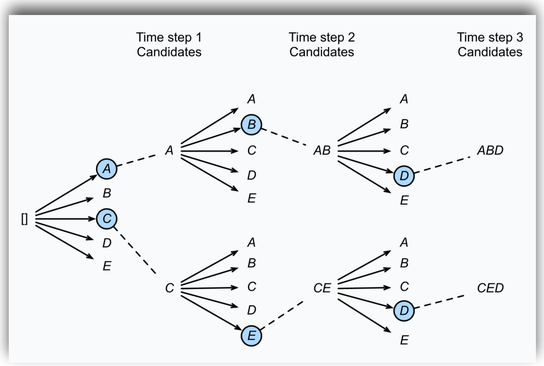
- The model produces decent results. However, there is a great scope of improvement, given adequate computational resources.
- Another area which needs focus is the inherent bias learned by the model. Due to the nature of Reddit (the source of data), the model sometimes produces inappropriate results. Due to this reason, we decided to not make the model publicly available yet.

Works Cited

- Sample Efficient Text Summarization Using a Single Pre-Trained Transformer, arXiv:1905.08836v1 [cs.CL].
- The Illustrated GPT-2, Jay Alammar.
- Stanford CS 230 - Deep Learning cheatsheet.
- Beam search article at d2l.ai

Sampling method

- At each time-step, the model assigns probability to what the next word could be. The way this word is chosen can be defined by some heuristic as defined by us.
- Several sampling methods exist such as the very simple greedy search, top-k or top-p sampling, and beam search.
- We experimented the trained model by using each one of these options and found that greedy search produced poor replies, with almost always repetitive contents; top-p and top-k sampling produced somewhat better replies, but still not satisfactory enough. Ultimately, we chose beam search as the final method of reply generation.
- Due to this choice, a certain amount of randomness is also induced in the replies and they're not the same each time.



- Depiction of beam search with a beam width of 2 and maximum output sequence length of 3.
- The candidate output sequences are A, C, AB, CE, ABD, CED.
- Out of these, the one with highest cumulative (conditional) probability is chosen as the final output sequence.