LightGAN: An Adversarial Approach to Natural Language Generation at a Large Scale

Jonathan Booher (jaustrinb), Nithin Kannan (nkannan), Enrique De Alba (edelba)
Department of Computer Science

Motivation
Natural language generation is increasingly important in today’s world of digital assistants. It is, however, difficult to have these systems produce language that makes sense. Traditional approaches like n-grams suffer from repeating corpus text and RNNs suffer from poor scaling as the vocabulary increases.

We therefore present a method that we call LightGAN. A GAN trained with a novel LSTM design originally from Microsoft that can address large vocabularies with minimal space requirements.

Data
- 467 Million tweets from 2009 from the SNAP group [1]
- Example raw data:
  T 2009-06-30 23:59:51
  H http://twitter.com/eboe

Preprocessing
- Remove timestamps and user information
- Remove non-English language tweets
- Replace websites, emojis, and @s with special tokens
- Pad the lines to the max length and remove words that appear less than 5 times.
- All preprocessing done beforehand to ensure that is not the bound
- Reduced vocabulary size to 100,000
- Example processed data:
  Out for karaoke and shots. Text if you dare. </url> </naw> <naw> ... <naw>

Key Idea:
- Allocate words into a 2D table
- Learn embeddings for each column and row
- A prediction for a row and column is a prediction for a word.
- Redistribute periodically to group similar words together in rows

Savings:
- Table allows us to perform two softmaxes to ceil(sqrt(|V|)) instead of one to |V|
- Space savings of O(sqrt(|V|))

Method: LightRNN [2]

Word Allocation:
- Initially random
- Redistribute the words by solving a min cost max flow problem
- Have costs be proportional to the perplexity the model achieves on that word

Results
Training was implemented using ‘Curriculum Training’. Where the GAN is trained on increasingly large sequences [3]. Testing was accomplished using Beam Search with a beam width of 100.

RT <AT_TAG> what wud you do
RT <AT_TAG> gdi fastfood
<AT_TAG> continually crazy
<AT_TAG> be oversleeping my school

Discussion
- Size of the dataset causes computability problems
- Attention and dropout in the generator greatly improved the stability of the model
- Model still has problems with longer sequences

Future Work
- Compare these results to those produced by a gan using traditional LSTM
- Train on different vocabulary sizes to see if the scaling affects accuracy
- Improve stability by working with different schedules for D and G

References

Method: The WGAN-GP Language Model

The Language Model:
- Frame as supervised learning problem: predict the next word
- Use RNNs for sequence prediction
- Pretrain the embeddings and word allocation table

The WGAN [4]:
- Minimize the distance between the real and fake distributions
- Improves the stability of traditional GAN
- Use same architecture for generator and discriminator

The GP [4]:
- An improved format of gradient clipping for GANs
- Penalize the gradients for being far from unit length

Method: The WGAN-GP Language Model

The Language Model:
- Frame as supervised learning problem: predict the next word
- Use RNNs for sequence prediction
- Pretrain the embeddings and word allocation table

The WGAN [4]:
- Minimize the distance between the real and fake distributions
- Improves the stability of traditional GAN
- Use same architecture for generator and discriminator

The GP [4]:
- An improved format of gradient clipping for GANs
- Penalize the gradients for being far from unit length

Discriminator Loss (Orange) and Generator Loss (Blue) vs. Iteration

References