Comparison of Deep Learning Text Generation Models Trained with Song Lyrics

NATHAN STONE & ZACK STRATHE

CIS 732 FINAL PROJECT

Introduction (1/3)

For intelligent-systems to interact with humans, natural language is the method accessible to most of the world's population

State-of-the-field in text generation tasks is undoubtedly large language models (LLMs)

- Current LLM models like GPT-3 are trained to have hundreds of billions of parameters

Introduction (2/3)

Large language models can have problems with bias, as well as environmental concerns

- Bias comes from large training data set, which contains biased viewpoints that may end up in model output

While LLMs can be used to fine-tune to a specific corpus, we instead will utilize smaller language models to avoid negative aspects of LLMs.

Introduction (3/3)

We trained neural network models with a corpus of song lyrics to evaluate with unconditional text generation

- Baseline model:
  - Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM)
- Evaluation models:
  - RNN with LSTM (tuned hyperparameters)
  - Generative Adversarial Network (GAN) models:
    - SeqGAN
    - TextGAN
    - LeakGAN

Background & Related Work (RNNs)

Recurrent Neural Network models for text generation

- RNN-based models used for text generation because they use sequential data
- Can have problems with “vanishing gradient”
  - Long short-term memory (LSTM) cells remedy the vanishing gradient
- RNN-LSTM has been used previously, in a similar task to ours, to “ghost write” lyrics
- RNN-LSTM models can still suffer from “exposure bias”
  - Exposure bias occurs because model was trained on a sequence of words, but when trained model is used for inference, the only input to the model is its own generated sequence
  - Leads to deterioration of generated text as sequence length increases

Background & Related Work (GANs)

Use of GANs in text generation emerged to alleviate the exposure bias inherent to RNN models

GANs consist of two sub-models:

- Generator – generates synthetic data
- Discriminator – evaluates synthetic data to determine if it is real or fake

In image generation, GANs have been very successful

- Can generate photorealistic images that are completely synthetic

In text generation, GANs have been less successful

- Because text generation deals with sequences of discrete tokens, while image generation is with continuous, real-valued data which makes propagation of model parameter weights much simpler

Text generation GANs utilize aspects of reinforcement learning to overcome weight propagation issue

Background & Related Work (SeqGAN)

SeqGAN trains a stochastic parameterized policy via a policy-gradient method

Generator is a RNN with LSTM cells

Discriminator is a convolutional neural network (CNN) to classify examples as real or fake

Generator uses a Monte Carlo search to approximate state-action values

Reward value for the generator is the likelihood that it will trick the discriminator

After pretraining to initialize the generator and discriminator model weights, they are trained adversarially

Background & Related Work (TextGAN)

The authors of TextGAN motivated to solve “mode collapse” with text generation GAN models

- When a generator tends to produce a single output for multiple representations of encoded text

Use a feature-matching approach to train the generator

- Instead of attempting to trick the discriminator, the generator is instead trained to more-closely match an encoding from the discriminator

Generator is a RNN with LSTM cells

Discriminator is a CNN

The generator deterministically outputs the first word of a sentence using the encoded feature-vector from the discriminator, then the remaining tokens are sequentially generated by the RNN

Background & Related Work (LeakGAN)

Authors of LeakGAN noted that research with GANs has not focused on longer text generation (>20 words)

- Binary signal from discriminator is too sparse because it's only generated for a full-length sequence
- And the signal is not informative enough for the generator to learn for longer sequences

LeakGAN utilizes elements of hierarchical reinforcement learning

- Generator is split into a high-level "manager" and a low-level "worker" module

Generator modules both use LSTM networks, while discriminator uses a CNN

Discriminator serves primarily as a feature-extractor, and the last-layer of the CNN is "leaked" to the generator

- "Manager" receives the feature vector and outputs a goal vector to the "worker"

Methodology (Computing Platform)

To train deep neural networks, need to utilize GPU for parallel computations and reasonable model training times

- Using Google Colab for free GPU access

Due to Google Colab time limitations, we had to scale-back the training corpus size and text generation length

- Generating sequences of 10 words maximum

Methodology (Models)

Baseline RNN
- Keras sequential model with LSTM, Dense layer
- Simple implementation with with simple parameters

Improved RNN
- Keras sequential model with LSTM, dropout layer, dense layer
- Tuned parameters
  - More LSTM units
  - Temperature
  - Additional epochs

GAN Models
- Utilizing TextBox module for python
- Simple implementation of custom training data
- Implements a number of GAN models for text generation
- Modified slightly to save training loss logs to .csv files

Experiment Design (Data Set)

Song lyrics corpus consists of lyrics selected from 8 artists:

- Stevie Wonder, David Bowie, Tool, Nine Inch Nails, Metallica, Black Sabbath, Jay-Z, and Frank Zappa
- Lyrics for 31 songs randomly selected from each (248 songs total)

Corpus split 50/50 for training and evaluation

- Duplicates removed to ensure that each set is unique
- Corpus split sequentially to ensure that training and evaluation sets contain a roughly equal distribution from each musical artist and song
  - So that word choice and themes from the text are similar in both data sets

Unique words: 6,411

- Training set: 4,317 unique words
- Evaluation set: 4,294 unique words
- Potential issue: approximately 1,050 different unique tokens between training and evaluation sets (approximately 24% of each) – could lead to poor evaluation performance, but shouldn't affect quality of generated text

Experiment Design - Evaluation Methods (1/2)

Primary evaluation: human scored

- Due to limitations with computer-generated scores to account for meaning and sentence structure

Human evaluation metric: scored 1-5 based on a combination of meaning and grammatical correctness:

- **1**: example contains no real words
- **2**: example contains some real words but lacks structure or meaning
- **3**: example contains some real words and has some meaning (such as, contains a subject and a verb)
- **4**: example contains all real words, but lacks some coherence and meaning
- **5**: example contains all real words, and seems like it could be a real song lyric

Important note: there is no length requirement for a song lyric, so an example with just one or two words could receive a score of 5

- Examples from evaluation set of single or two-word lyrics: “Attack”, “Right”, “Goblin girl”

Experiment Design – Evaluation Methods (2/2)

Secondary evaluation (in case of tiebreaker for human evaluation):

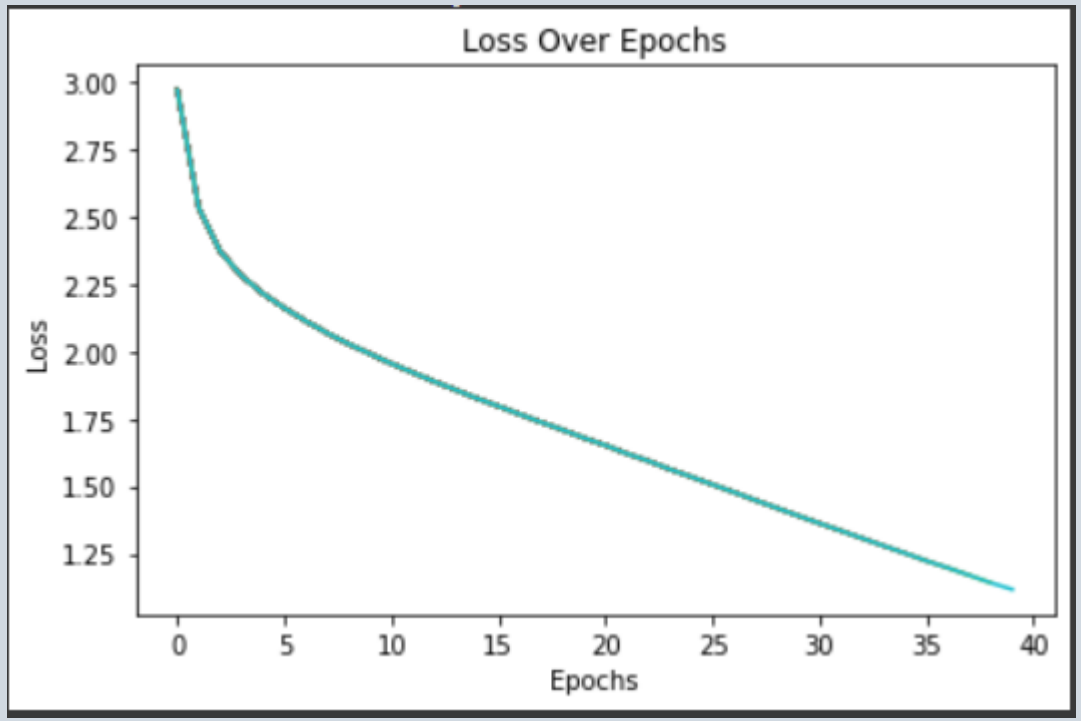
- Bilingual Evaluation Understudy (BLEU) scores
- Calculates the n-gram similarity between a sentence and a reference corpus

Calculating BLEU-1, BLEU-2, BLEU-3, BLEU-4, but utilizing only BLEU-3 for evaluation

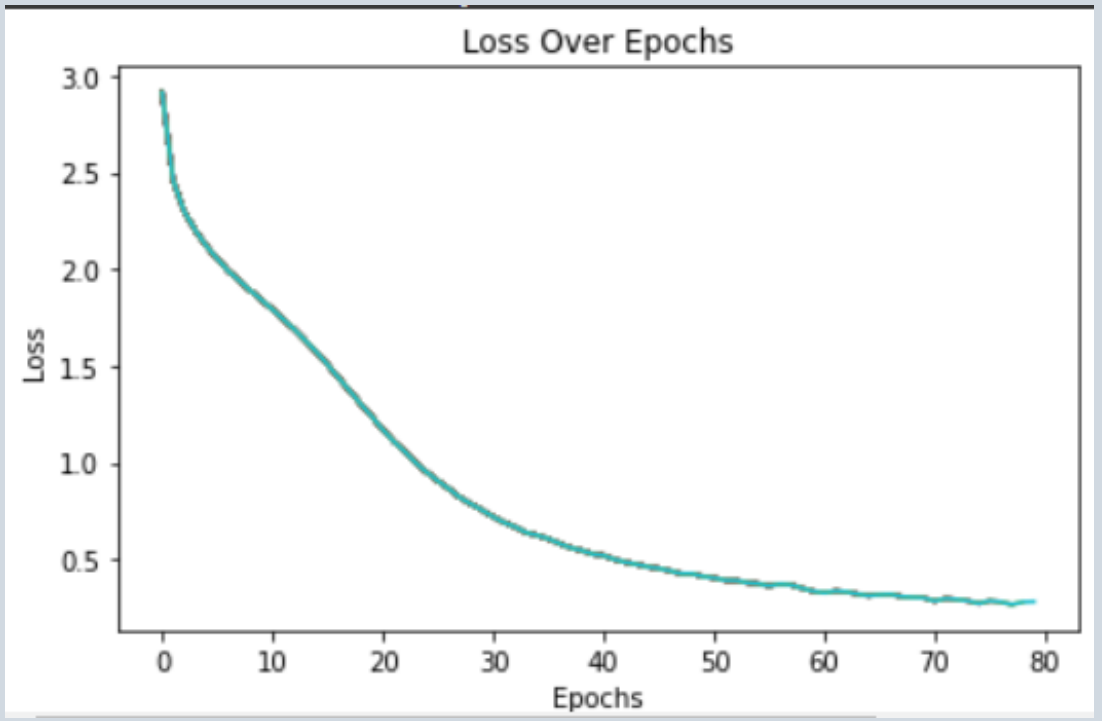
- We believe evaluation of 3-grams is ideal since models are trained with a sequence length of 10 words:
  - 2-grams may not be able to capture semantic relationships well enough
  - 4-grams may be too sparse within the training and generated texts to achieve a meaningful comparison between models

Results (1/3)

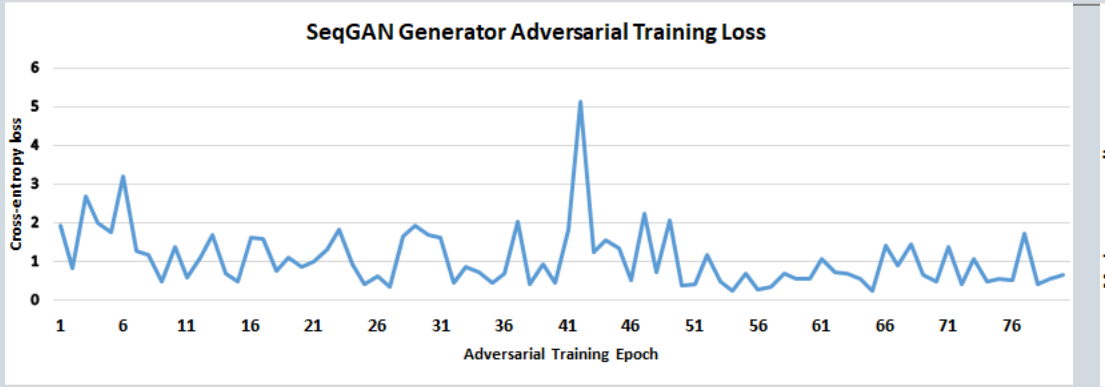
Baseline RNN with LSTM

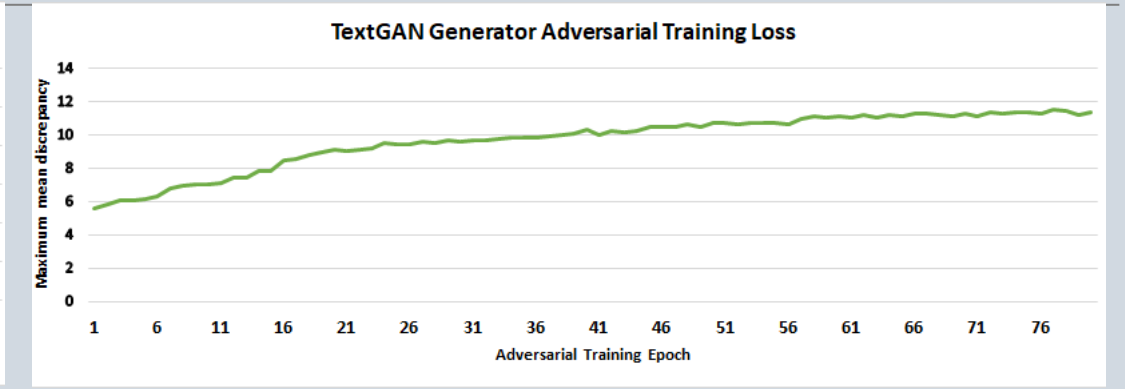


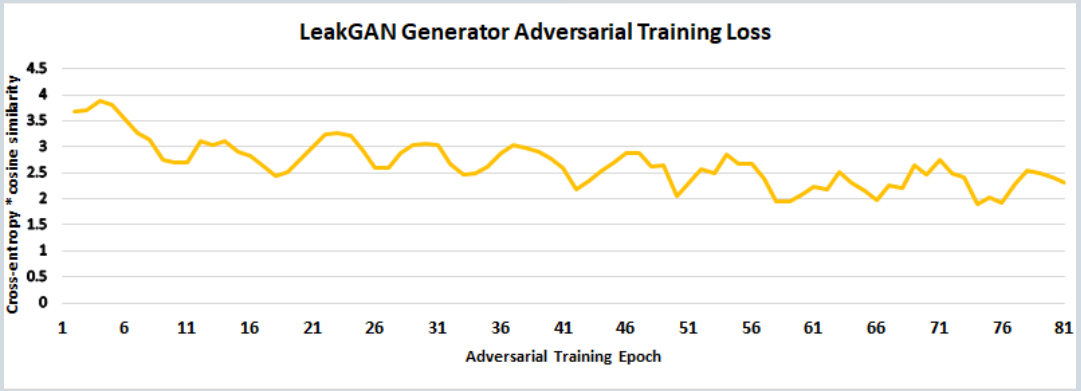
Adjusted RNN with LSTM



Results (2/3)







SeqGAN attempt to minimize cross-entropy loss, while TextGAN utilizes the maximum mean discrepancy metric, and LeakGAN utilizes a cross-entropy* cosine similarity loss for adversarial training

Results (3/3)

Model	Human Eval	BLEU-1	BLEU-2	BLEU-3	BLEU-4
RNN (baseline)	1.55	0.415	0.074	0.038	0.046
RNN (tuned)	2.80	0.744	**0.266**	**0.058**	0.045
SeqGAN	**4.55**	0.786	0.176	0.056	**0.063**
TextGAN	3.90	**0.811**	0.187	0.046	0.049
LeakGAN	3.85	0.802	0.227	0.045	0.034

In regard to the human evaluation, the GAN models seemed to perform better than both the baseline RNN and the RNN with tuned hyperparameters

BLEU scores seem to be less conclusive, as the better performing model varies with the n-gram similarity evaluation.

Generated Text Examples (1/2)

RNN (baseline)
folling world,
as a 1selly murise freez labming ah please halls skirsendled
a foor the marst
your mocawint on bling the 'reer
ift cruplifes lough
oh by he vererestiy
which see beending west is...., in't cauneed undi and yseeay

RNN (adjusted)
don't let me to live to like i take you
please the ruck if the stare
the chused and shear
here a must bord i'm on a pellong
and i was the way your bay you've got
ther that they cruice from me
what you get me to be

Generated Text Examples (2/2)

SeqGAN	and we draw fire
	to feel it like and turn out
	hate
	and kill his joy
	chase back
	listen of love
	put me when i know the family

TextGAN	it 's just a fella ?
	the show the potential
	well it belongs my money ?
	y'all worries them burden
	refugees of crushin , lovers
	hands bags through me now
	wrap the damage when and listen to fight away up

LeakGAN	i want to date i got my back
	you ca n't got ta claim if you
	well , she hid down clouds midnight hot
	i 'm in cuba , love , still get , surprise
	i 'm not such a stranglehold , i 'm down
	i 'm eating
	i 'm not mind girl

Conclusion

GAN models outperformed the RNNs with LSTM in human evaluation

GAN models did not seem to perform significantly better with BLEU scores

- BLEU scores may not be a great evaluation metric for generated text
  - Does not account for sentence meaning or structure

SeqGAN seems to be the best text generation model of those that we evaluated

Future Work

Re-evaluate models but collect training times to compare computing resource requirements for each model

Utilize a larger corpus and longer text generation length, which could improve the quality of results

Further parameter adjustment to each model using exhaustive grid search