deephypebot: Music Commentary Generation with Deep Learning

Nadja Rhodes [1] **Natasha Jaques** [2 3]

Abstract

In this paper, we describe a deep generative language model trained on past human music writing, compiled from the web and conditioned on attributes of the referenced music. After pre-training a conditional sequence-to-sequence variational autoencoder (VAE), the model is refined using a latent constraints generative adversarial network (LC-GAN). This additional training step makes use of latent Dirichlet allocation (LDA) topic modeling, to encourage the model to output descriptive, almost flowery writing commonly found in this style of music commentary. Our goal is to teach this language model to generate consistently good and entertaining new writing about songs. The models outputs are deployed live via @deephypebot[1], a Twitter bot that automatically finds ongoing discussions about music and generates new music commentary.

1. Introduction

In recent years, there has been a steady rise in interest in applying deep learning methodologies towards natural (i.e., human) language processing and understanding (*NLP/NLU*). Applications such as text classification, machine translation, and reading comprehension all stand to benefit from such efforts.

One growing school of thought is that, borrowing from successes in the computer vision community, transfer learning using pre-trained language models holds an important key to improving the state of the art in deep NLP[2]. *Language models* attempt to predict the next word (or character, or any linguistic unit) in a sequence by estimating its probability distribution given the previous sequence of words. There has been healthy debate, however, around the limitations and abilities of such methods to truly understand language[3].

This debate is at the heart of state-of-the-art NLP research today, where the most popular model architectures are long short-term memory networks (LSTMs) and the Transformer. This project aims to take a more exploratory journey through the elegant concepts of the variational autoencoder (VAE) architecture and the increasingly popular generative adversarial network (GAN). VAEs and GANs are both prominent generative network architectures frequently used for non-textual creative applications. Examples include SketchRNN (Ha & Eck, 2017), MusicVAE (Roberts et al., 2018), and neural image translation like pix2pix (Isola et al., 2017).

VAEs and GANs are less commonly applied toward language modeling. Aside from its potential utility for downstream tasks as discussed earlier, language modeling is a fascinating creative task in its own right. A model that can accurately generate language yields an exciting potential: machines that can *write*.

Using an approach which combines a conditional sequence-to-sequence VAE language model with a GAN that can refine and improve its output, this project attempts to train a model to automatically generate creative music commentary. Specifically, we strive to train the model to write about songs in a way that is topical, structured, and specific to attributes of the referenced song. The project culminates in a Twitter bot that automatically finds tweets about current songs and generates commentary about them, disseminating the model outputs to the wider community.

To the best of our knowledge, this project is the first to apply this type of GAN and VAE approach to text generation, as well as the first to examine automatic music commentary generation using deep language modeling.

2. Music Commentary Data

The language around music – manifested by hundreds, even thousands, of "nice, small blogs" on the internet – has fueled the discovery of new music among enthusiasts for years, particularly in the late 2000s. This project ultimately aims to pay homage to these deep and unique wells of creative writing.

Hype Machine (HypeM) is a music blog aggregator. It was recognized (e.g., by VICE Magazine) as an internet tastemaker at the height of its popularity, and few projects

[1]OpenAI Scholar [2]MIT Media Lab [3]DeepMind. Correspondence to: Nadja Rhodes <narhodes1+deephypebot@gmail.com>.

[1]https://twitter.com/deephypebot

[2]https://thegradient.pub/nlp-imagenet/

[3]https://bit.ly/2PxoCOG

capture the energy of this community quite like its "eclectic stream."

With the paid HypeM developer API, we retrieved ˜10,000 unique songs from the site's weekly popular charts, over the last 5 years. From this song list, we branched to the urls of blog posts discussing each song, producing ˜80,000 urls of written music commentary. Figure 1 shows the distribution of word counts per piece of commentary.

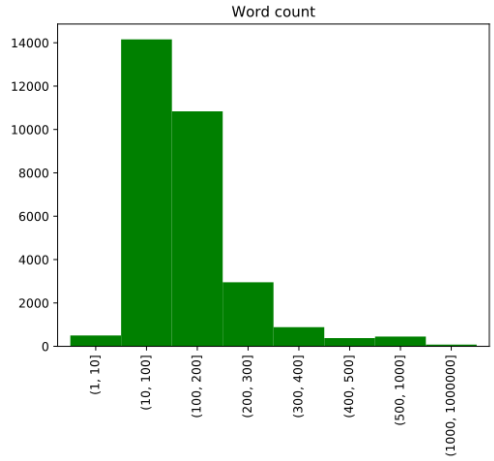

Figure 1. Word count distribution per music commentary piece. The mean word count in a piece is 132, and there are almost 4M total words.

This initial ˜80K count eroded with data cleaning: posts that highlighted multiple tracks in a single "roundup"-style post were removed, many urls no longer existed (even on Web Archive) and could not be retrieved, and we enforced a policy that the commentary should be in English about a song that could be found on Spotify (to allow for the retrieval of interesting attributes related to the song). Figure 2 shows how the number of pieces of commentary funnels down as we apply these filters, to an eventual ˜30K pieces.

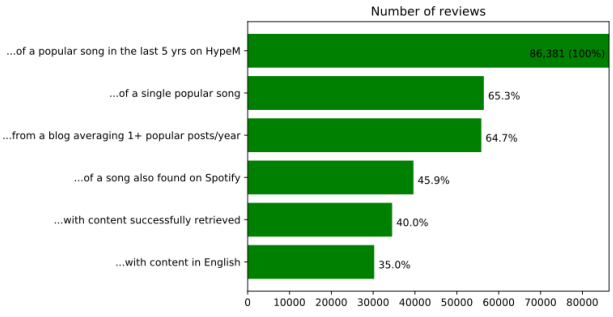

Figure 2. Data loss in music commentary dataset.

To ensure data quality, we did additional manual cleanup of symbols, markdown, and writing that was deemed non-commentary. From there, the commentary was split into ˜104,500 sentences – sentences being a good length for a variational autoencoder (VAE) model to encode.

2.1. Sources of bias

Like any brand or outlet, Hype Machine has a certain perspective on the music landscape. As a music blog aggregator, it is a convenient data source for this project; yet in relying exclusively on HypeM, we make a value judgment that its blog list represents the most valuable type of music writing on the internet (and thereby exclude, for example, social music writing on platforms like Tumblr or SoundCloud). Conversely, we are largely assuming that all writing from this blog list is valuable.

Hype Machine is a tastemaker – but then, what is taste? For example, The Chainsmokers, an electropop duo, accredited HypeM for catapulting their careers in the same profile in which they described themselves as "frat bro dudes." So whose tastes are being left out?

Take, for instance, SoundCloud rap, which is having a big mainstream moment right now. What happens if we (roughly) compare the performance (as of 7/21/2018) of The Chainsmokers on HypeM to that of a Lil Uzi Vert, who has successfully transitioned from SoundCloud to the top of the Billboard charts? Tables 1 and 2 show the relevant data per artist. The Billboard numbers are used here as an indicator of mainstream popularity, and The Chainsmokers and Lil Uzi Vert are fairly comparable. Yet The Chainsmokers have 2.3x more tracks listed on HypeM (including remixes by other artists) than Lil Uzi Vert. Granted, many factors can influence HypeM chart performance – but the EDM-pop Chainsmokers do seem a bit over-represented on HypeM, compared to the mainstream. This is undoubtedly intentional (the HypeM community can certainly have its niche), but awareness of skews in the dataset which is used to train the model is important for any project.

The Chainsmokers
Billboard Hot 100: **1** #1 hit, **5** Top 10 hits, **12** total songs
HypeM: **97** tracks (**188,696** times loved by community)

Table 1. The Chainsmokers: Billboard & HypeM performance

Lil Uzi Vert
Billboard Hot 100: **1** #1 hit, **2** Top 10 hits, **23** total songs
HypeM: **42** tracks (**3,129** times loved by community)

Table 2. Lil Uzi Vert: Billboard & HypeM performance

As further evidence, we compare with another rap group:

Migos. This group has better Billboard numbers, but they still underperform The Chainsmokers on HypeM, as shown in Table 3.

Migos
Billboard Hot 100: **1** #1 hit, **4** Top 10 hits, **32** total songs
HypeM: **82** tracks (**17,322** times loved by community)

Table 3. Migos: Billboard & HypeM performance

In general, pop-leaning tracks tend to dominate the HypeM popular charts. According to a genre-labeling scheme obtained from Genius.com, the single-genre breakdown of the commentary dataset taken from the last 5 years on Hype Machine is given in Table 4. Note that it skews heavily towards pop music.

Genre	**Percent**
Pop	64%
Rap	15%
Rock	14.5%
R & B	6.3%
Country	0.2%

Table 4. Genre composition of music commentary dataset

We also detected anecdotal evidence of bias in the kinds of samples the models have been most willing to generate:

- Bias towards certain locations, e.g.,"Berlin-based producer" or "LA-based producer."
- Bias towards male-gendered pronouns, e.g., "*his* soulfully introspective" or "his debut/take/new track."

While this is a high-level analysis, we think it emphasizes an important point: there are many opportunities for discrimination in deploying machine learning systems[4], and it is important to be self-critical as a machine learning practitioner. It is not difficult to discern patterns of *selection bias* in this dataset.

2.2. Genres

The Spotify Web API[5] provides access to the metadata of millions of songs, artists, and albums in the Spotify catalog. Two metadata types of particular interest for this project are artist genres and audio features. We hypothesized that conditioning generated commentary by providing such properties of the referenced music at both training and inference time could provide the specificity we desired in the writing.

Spotify genres are so amazingly specific that they can be considered *micro*genres. With labels like "vapor twitch," "indie poptimism," and "stomp and holler," these new genres go far beyond the bland and conventional labels of pop/rap/rock/r&b/country from the previous section. These genre labels tend to convey a mood, as expressively and concisely as possible. Figure 3 shows a word cloud of the top genres detected in our music commentary dataset.

This kind of specificity, as well as combinatorial power (as multiple genre labels could be assigned to a single artist), brought about concerns that conditioning on such information could isolate certain writing, allowing the model to effectively memorize the training data; a problem which is heightened by the limited size of the dataset. Since memorization is antithetical to generalization, this could greatly restrict the creativity of the model. Imagine a request for commentary on a "chamber pop/chillwave/neo-psychedelic/nu gaze" song came in to the model, but the model has never seen such a genre combination before – what will it do? This turned out to be a very relevant concern.

After some simple clustering in an attempt to somewhat mitigate the long tail of the genre distribution, we obtained 130 unique genre labels over almost 20,000 pieces of commentary (note the further reduction in data from 30K previously – this is due to occasions where Spotify had a song but did not have related genre information). There are 1,135 unique genre combinations, since each piece can contain multiple genres.

When we started examining and clustering genres, many neat insights emerged, which underline the discoveries made in the bias investigation in the previous section. The Every Noise[6] project enabled a deeper dive into how to better interpret these somewhat opaque genre labels. It provides "an algorithmically-generated, readability-adjusted scatterplot of the musical genre-space," where "in general down is more organic, up is more mechanical and electric; left is denser and more atmospheric, right is spikier and bouncier." It can tell you what it believes is the "most representative" song for a genre label. It links to genre-based playlists on Spotify. For samples of the most representative songs from genres present in the dataset under investigation, please see: https://iconix.github.io/dl/2018/08/14/project-notes-1#data-exploring.

Every Noise contains scatter plots to show both similar and dissimilar genres to a specific genre. This gave us the insight that the top 11 genres in the dataset are considered very similar to each other. Also similar: 13, 15, 17, 23...

[4] https://nlpers.blogspot.com/2018/06/many-opportunities-for-discimination-in.html

[5] https://developer.spotify.com/documentation/web-api/

[6] http://everynoise.com/

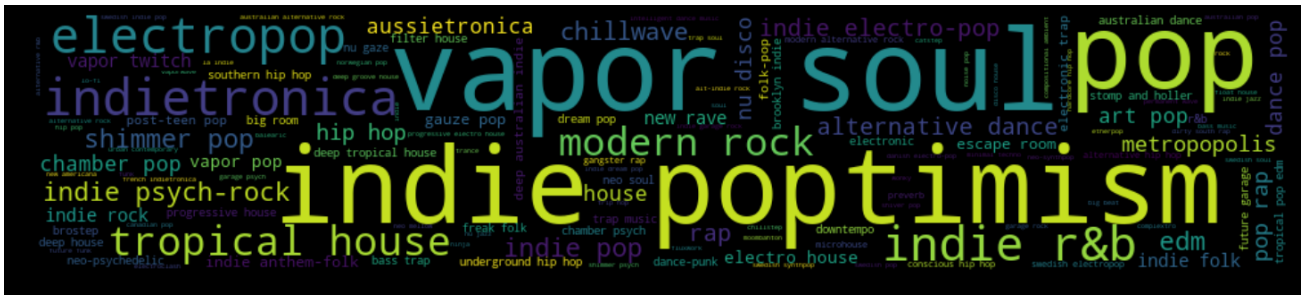

Figure 3. Top Spotify microgenres in the collected music commentary dataset.

this again speaks to selection bias in the dataset. The top genres for our music commentary dataset are given in Table 5. Some bias-related insights to highlight:

- *The #1 genre found in the commentary collection?* Vapor Soul.
- *How different is the #1 from the #2 genre, Indie Poptimism, according to Every Noise?* Not very.
- *Which genre that was at least visible (at 0.2% representation) in the previous genre labels is conspicuously missing now?* Country.

Microgenres	Piece Count
vapor soul	4344
indie poptimism	3886
pop	3799
indietronica	3462
electropop	3406
indie r&b	3242
tropical house	2359
modern rock	2171
indie psych-rock	1942
indie pop	1845

Table 5. Top 10 Spotify microgenres in commentary dataset

3. Machine Learning Methods

A *language model* (LM) is an approach to generating text by estimating the probability distribution over sequences of linguistic units (characters, words, sentences). This project centers around a conditional sequence-to-sequence variational autoencoder (seq2seq CVAE) that generates text, which is then refined with an additional latent constraints generative adversarial network (LC-GAN). The LC-GAN helps control aspects of the generated text by incorporating knowledge of the text topics, which are obtained by training a latent Dirichlet allocation (LDA) topic model on the music commentary data.

At a high level, the seq2seq VAE consists of an LSTM-based encoder and decoder. Once trained, the decoder can be used independently as a language model conditioned on latent space z. Then the LC-GAN can be used to fine-tune the input z to this LM to generate samples with specific attributes, such as realism, readability, or the desired topic distribution as modeled through LDA. It can also be used to condition generations on attributes like song audio features or artist genres.

3.1. Sequence-to-Sequence Variational Autoencoder

Sequence-to-sequence (seq2seq) (Sutskever et al., 2014) describes a type of deep neural network designed to handle the case where both the input and output of the network are variable-length sequences. It consists of one RNN that encodes the input sequence, then hands its encoding to another RNN which decodes the encoding into a target sequence; this architecture is shown in Figure 4.

The intermediate encoding vector z is known as the thought vector (and also the embedding, hidden code, representational vector, or latent vector), and is a compressed representation of the knowledge that was needed for the encoder-decoder network to translate from input to target. The goal is to train the model to capture all relevant factors of variation in the data using the latent code or thought vector.

An *autoencoder* is a technique in which the same data is used as both the input and target output for an encoder-decoder network. To prevent the network from simply learning the identify function, the embedding layer is constrained to have much lower dimensionality than the data. This bottleneck forces the network to compress any relevant information needed to reproduce the data. Thus, in the network's attempt to learn to copy the input to the target, the thought vector z will become a latent (hidden) representation of the data provided.

A *variational autoencoder* (VAE) (Kingma & Welling,

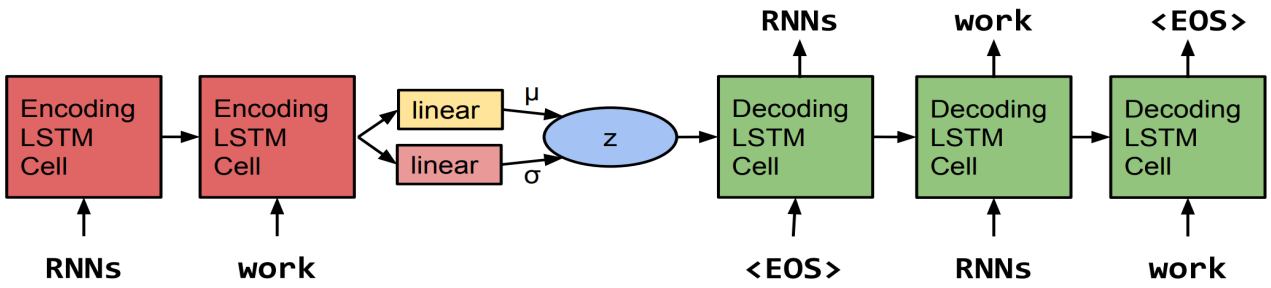

Figure 4. Seq2seq VAE architecture. Replace z with $\langle z, a \rangle$ for a CVAE. Figure taken from Bowman et al. 2015

.

2013) is a modification to the autoencoder which constrains the distribution of thought vectors to resemble a unit Gaussian. Specifically, rather than producing a single thought vector z, the encoder outputs a mean $\mu(X)$ and variance $\Sigma(X)$ based on the input X. Using the reparameterization trick (Kingma & Welling, 2013), these are used to sample a $z \sim \mathcal{N}(\mu(X), \Sigma(X))$, which is the new latent vector. A KL-divergence term is added to the loss to constrain the distribution over latent codes to resemble a unit Gaussian:

$$D_{KL}[\mathcal{N}(\mu(X), \Sigma(X))||\mathcal{N}(0, \mathbb{I})] \quad (1)$$

This gives the VAE the important ability to generate novel samples of the data it is modeling. One can simply sample a random z from the unit Gaussian, $z \sim \mathcal{N}(0, \mathbb{I})$, and feed this into the decoder to generate a novel sample.

In order to provide the decoder with additional information, it is possible to condition the VAE on additional variables a, by concatenating them with the thought vector z, and using this as input to the decoder. Thus, the decoder becomes a function of $\langle z, a \rangle$. We use this approach to condition our decoder on additional information about the song, making our model a conditional VAE (CVAE). This allows us to generate opinionated music commentary that is tied to the characteristics of the song being reviewed; we can find a tweet containing a song, query the Spotify API to obtain its genre and other attributes a, sample a random z, and feed $\langle z, a \rangle$ into the decoder to get an opinionated review conditioned on these attributes.

Putting this all together, the seq2seq CVAE is a network that can first learn a rich latent representation of the text, and then use that representation to generate new samples, informed by the attributes of the song. This model is ideal for this project, because it allows for the generation of novel music commentary through simply collecting some information about the song, sampling a random z from the latent space, and feeding this into the decoder. Further, it may be more likely to generate topical, coherent text, because the thought vector must represent global properties of the text. Using it to generate commentary should incorporate more general, abstract knowledge than a simple LSTM-LM, since the LSTM-LM can only generate language by making local, word-by-word predictions. These predictions are limited by the capacity of the LSTM, which is known to be poor at retaining high-level information in long sequences.

However, even the seq2seq VAE cannot overcome the difficulties of generating interesting, coherent samples of natural language. VAEs also suffer from the inability to model long sequences; as Bowman et al. (2015) state, "As the sentences get longer, the fidelity of the round-tripped sentences decreases." Secondly, any approach to language modeling trained with maximum likelihood estimation (MLE) (*i.e.* estimating the probability of the next linguistic unit based on the training data) is inherently limited. LMs trained with MLE tend to generate text that is dull, generic and repetitive (Li et al., 2017). This has led several researchers to consider approaches to text generation that first pre-train a LM using MLE, then refine it using RL or other approaches (e.g. (Li et al., 2017; 2016). We take a novel approach by applying a GAN to refine our seq2seq VAE model, as described in the next section.

3.2. Latent Constraints Generative Adversarial Network

A *generative adversarial network* (GAN) is a neural network comprising two models: a generative model G that attempts to model the distribution of the data, and a discriminative model D that estimates the probability that a sample came from the true data, rather than G (Goodfellow et al., 2014). In other words, the generator G is trained to fool the discriminator D into thinking that the samples it produces are real, while D is trained to learn how to distinguish between real and fake (G) samples. In this way, the two models are adversaries.

A *latent constraints* GAN (LC-GAN) provides a way of conditioning the generations of a pre-trained VAE using

a GAN (Engel et al., 2017). Engel et al. (2017) describe latent constraints as "value functions that identify regions in [VAE] latent space that generate outputs with desired attributes." In other words, the LC-GAN is able to improve a VAE by identifying the regions of the latent embedding space that lead to the generation of good samples. The latent constraints enable a user to fine-tune a VAE to generate samples with specific attributes, all the while preserving each sample's core identity. Note that both VAEs and GANS are considered deep latent-variable models because they can "learn to unconditionally generate realistic and varied outputs by sampling from a semantically structured latent space" (Engel et al., 2017). This structure is what the latent constraints aim to manipulate towards more 'satisfying' samples.

Since adding latent constraints is a fine-tuning procedure it has important benefits: 1) no retraining is required, because the weights of the original VAE remain the same, and 2) compared to training from scratch, a much smaller labeled dataset (or rules-based reward function) is needed. For example, even when using highly noisy labels based on facial expression responses, Jaques et al. (2018) were able to improve a sketching VAE using only ˜70 samples.

Following Engel et al. (2017), we use the LC-GAN to optimize for certain attributes exhibited by the samples. Essentially, we choose some value function V that we would like to control. The discriminator is then trained to output the value v of a given z vector. The (z, v) data used to train the discriminator can consist of hand-labeled samples, or v can be computed automatically on the samples generated by decoding z. In our case, the value function is obtained through the outputs an LDA topic model. We distill the LDA model into the discriminator, by training the discriminator to output the topic distribution for a given z. We train the generator to produce samples that favor a particular topic, which corresponds to a descriptive, almost flowery style characteristic of the highest-quality music commentary.

The LC-GAN training procedure and architecture is shown in Figure 5. The grey column in the middle of the diagram represents the training loop. An iteration consists of M batches of discriminator D training, followed by N batches of generator G training, where $M >> N$. We intentionally train D more than G because the generator is very sensitive to overtraining. This is because too much training can lead to G generating samples far from the Gaussian distribution expected by the decoder, leading to bizarre results.

The discriminator D is trained to estimate the value of random z vectors, $D(z) = \hat{(v)}$. The z is then decoded into a sentence, which is fed into the LDA topic model to contain the true value v. The cross-entropy between $\hat{v}$ and v is used to train the discriminator. Training the discriminator takes relatively few samples; as Jaques et al. (2018) state, "because the latent space of a VAE is already a small-dimensional representation, it is straightforward to train a discriminator to learn a value function on z, making the LC-GAN well-suited to learn from small sample sizes." This makes LC-GAN training fast (computationally inexpensive) and straightforward.

The $z \sim \mathcal{N}(0, I)$ input into G is "noise" from a random Gaussian (normal) distribution. The generator is meant to turn this noise into a z' that is able to produce a good sample. The z' output by G is then fed into D, which produces a label v that is then used directly as the loss to judge whether G indeed produced a sample that satisfy the desired attribute constraints.

The LC-GAN is an interesting approach that takes advantage of the complementary strengths and weaknesses of VAEs and GANs. GANs suffer from mode collapse, which occurs when the generator assigns only models a small portion of the true distribution, effectively missing out on generating a plurality of realistic samples. VAEs suffer from a blurriness trade-off. According to Engel et al. (2017), they can "either produce blurry reconstructions and plausible (but blurry) novel samples, or bizarre samples but sharp reconstructions" – this results in a compromise between good samples and good reconstructions. By using the GAN on top of the VAE, it is possible to train the VAE to optimizing for good reconstructions, and have the generator compensate for the poor sampling behavior by transforming a randomly sampled z into a z' that can be decoded into a good sample.

Therefore, when using the LC-GAN approach, the VAE should be trained to achieve good reconstructions, at the expense of good sampling. Good reconstructions ensure realism, sharpening sample quality without sacrificing sample diversity (Engel et al., 2017). Note that the generator G can be trained to preserve the identity of a sample by making the minimal adjustment in latent space to satisfy the discriminator; this is accomplished by adding a penalty to the loss for the distance in latent space between z and z', *i.e.* $||z' - z||$. This minimal transformation constraint allows the core identity of a sample to be preserved, even during attribute transformations. Further, it helps the GAN to avoid mode collapse and encourages diverse samples. This allows for a remarkable symbiotic relationship between a pre-trained VAE and a fine-tuning GAN.

The convenience of applying the LC-GAN may give the impression that applying GANs to NLP problems is easy, but this is not generally the case. Discrete sequences, like text, have a non-differentiable decoder: the output is sampled over a distribution of words. Gradients cannot be back-propagated through this sampling procedure, making it extremely difficult to optimize the output of a discriminator. The LC-GAN avoids this differentiation problem, because we are back-propagating the gradients through thought vec-

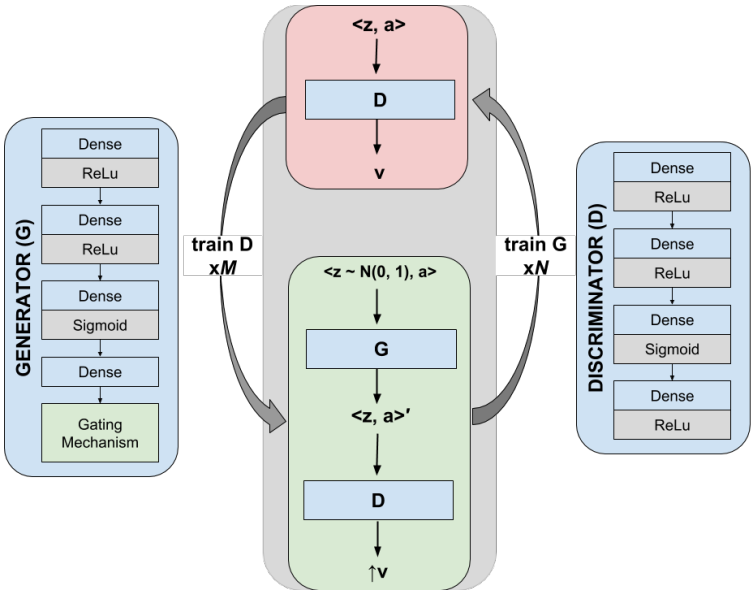

Figure 5. A diagram of the LC-GAN architecture. The center red box represents discriminator D training, and the center green box represents generator G training. The blue boxes along the sides show the internal architectures of G and D – both are similar, except the final layer of G is a gating mechanism that allows the generator to remember or forget what it wants about the $cat(z, a)$ it was provided as input and the $cat(z, a)$ it computed with its own layers.

tors z and not text.

4. Engineering Methods

This section will describe the engineering pipeline required to support inference live on Twitter, which is diagrammed in Figure 6. The code for this project has been open-sourced, and is available at `https://github.com/iconix/deephypebot`.

4.1. Making inference requests to the network

While the model was developed in Python (PyTorch, conda environment manager), the pipeline is written using Node.js in order to run server-side JavaScript. This is due to how many convenient libraries already exist on npm for communicating with external services. The pipeline has two main components: a worker process and a multi-endpoint Express.js web service.

The web service essentially wraps existing APIs for Google Sheets, Spotify, and Twitter, providing a more convenient set of APIs to the worker. The worker process monitors Twitter at an interval (currently every 60s) for new tweets to appear on @deephypebot's home timeline. New tweets are defined as any tweets that have occurred since the last tweet processed by this workflow (max as enforced by the Twitter API: 200 tweets). New tweets are parsed for song and artist information, and if this is found, the information is passed on to Spotify. If Spotify accepts and responds with genre information, this is then passed to the neural network (which is deployed behind a Flask endpoint) for conditioned language modeling.

4.2. From samples to tweets

Once multiple samples of commentary for a new proposed tweet are generated, they are added to a spreadsheet where a human curator can select which samples are released to @deephypebot for tweeting.

Text generation is a notoriously messy affair where "you will not get quality generated text 100% of the time, even with a heavily-trained neural network." While much effort will be put into having as automated and clean a pipeline as possible, some human supervision is prudent.

5. Results

5.1. Seq2seq VAE

To gain insight into the factors of variation encoded by the unconditional VAE model, we can interpolate between two points in the latent space. This is performed by encoding two sentences, s_0 and s_1, to produce their latent vectors, z_0 and z_1. We can then linearly interpolate between z_0 and z_1, decoding the intermediate representations, to see how the structure of the sentences changes as we move through latent space. These interpolations can be referred to as *homotopies*. As discussed in Bowman et al. 2015:

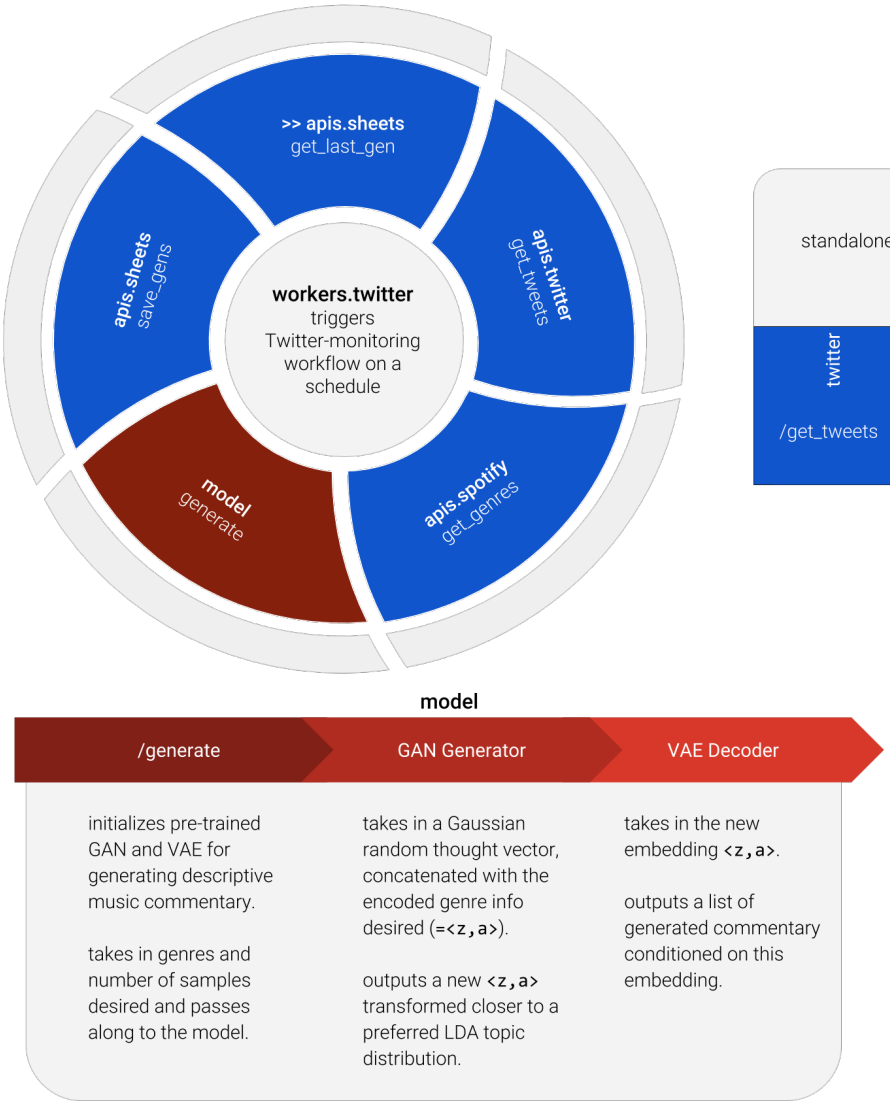

Figure 6. A diagram of the software architecture for deephypebot.

Examining these homotopies allows us to get a sense of what neighborhoods in code space look like – how the autoencoder organizes information and what it regards as a continuous deformation between two sentences.

See Table 6 for examples of generated homotopies. Note that conditioning makes the homotopies more brittle; thus, such homotopies are not included in this paper.

Homotopy sample #1
(s_0) *it had taken years to believe*
(z_0) *it categories hip hop for something*
(. . .) *it was last weekend and disco*
(. . .) *it was nothing but the weekend*
(. . .) *it was all weekend at the end*
(z_1) *it was it all weekend at the end*
(s_1) *but it was all lies at the end*

Homotopy sample #2
(s_0) *he was silent for a long moment*
(z_0) *i was working for a long moment*
(. . .) *i was hell for a long moment*
(. . .) *i was me for a long moment*
(. . .) *i was me in a moment*
(. . .) *it was one in my least*
(z_1) *did it song in my leave*
(s_1) *it was my turn*

Table 6. Interpolating between two sentences $s0$ and $s1$.

Demonstration of interpolating in the latent space is available at https://iconix.github.io/dl/2018/07/21/bias-and-space.

5.2. Seq2seq CVAE

Table 7 gives samples obtained by training a seq2seq CVAE that conditions its generations on genre information. These samples were chosen as the best from 50 random generations; as such, they represent the best-case performance of the model. Due to the richness of the genre information and the small size of the dataset, this can make the model more brittle, in that it is less likely to have seen data for a particular genre combination. However, it may be possible to correct this problem using the LC-GAN.

VAE samples show some subjective signs of genre influence. See training loss plot in Figure 7.

5.3. LC-GAN

Table 8 gives samples obtained by training an LC-GAN on top of the pre-trained seq2seq CVAE from 5.2. This helps

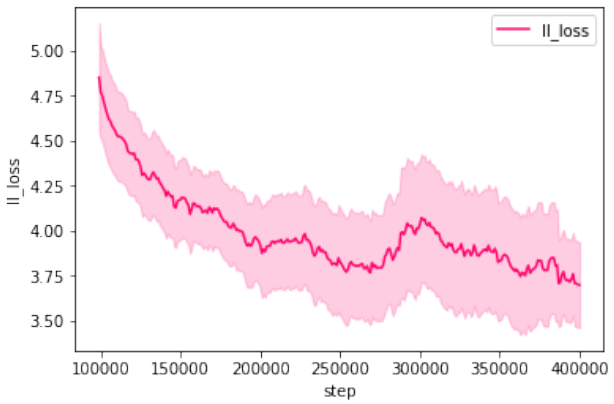

Figure 7. Training loss plot CVAE. Perplexity of 40.32 is indicative of the model overfitting to its training data.

optimize samples towards an ideal LDA topic distribution, providing more descriptive yet realistic samples. These samples were chosen from 50 random generations as representative of the capability of the LC-GAN to transform a VAE sample into a more realistic and/or descriptive sample; as such, they represent the best-case performance of the model. Ideal transformations are not attained, but one can discern some improvement via reductions in repetitiveness, reductions in "promotional" language, and occasionally more genre-influenced content. The basic VAE samples are also provided to demonstrate the transformation that the LC-GAN provided to the VAE samples. In other words, samples in the same row share the same $\langle z, a \rangle$.

5.4. deephypebot

Figure 8 shows a screenshot of samples automatically generated by the Twitter monitoring pipeline backing @deephypebot. Tweets represent a "best of" collection.

6. Discussion and conclusions

This project is the first to approach the problem of generating interesting and creative music commentary, through learning from a rich dataset of descriptive, eloquent writing about songs. The deep learning methods employed for the model represent a novel technique for creative text generation: refining a pre-trained seq2seq CVAE using a GAN and LDA topic model, to learn which aspects of the latent space lead to the ideal topic distribution. The model has been deployed live on Twitter, and is able to automatically detect tweets about songs, obtain information about the song's attributes, and feed this into the generative model to produce automatically generated opinionated comments about the song. We hope that this model will not only pay homage

Genre	VAE Sample
[‘big room’, ‘brostep’, ‘edm’, ‘electro house’, ‘electronic trap’, ‘moombahton’, ‘pop’, ‘tropical house’]	hes coming out of us that hes coming out of the world and step back into banger
[’indie anthem-folk’, ’modern alternative rock’]	this track builds and a combination soaring soundscape that UNK UNK builds with a happy vibe
[’chillwave’]	song in my these days and is a lot of nothing but it is a dance party

Table 7. Random VAE samples with their conditioning genre.

Genre	VAE Sample	GAN+VAE Sample
[’hip hop’, ’rap’]	and and has returned to the and and and and and and and and and this new UNK with a UNK UNK UNK UNK i hope	self described as part of the the UNK UNK manages to UNK returned with a UNK UNK with each light
[’aussietronica’, ’deep australian indie’, ’electropop’, ’indie r&b’, ’indietronica’, ’vapor soul’]	you might not be a in in which be in this when it comes from their own different tracks	it feels be be stripped long before but it like comes in addition to its warm material
[’indie poptimism’]	listen to this song with the the vocals to as well as UNK UNK on new cut	lifted from the the above and listen to the sound as well as a spin on new cut

Table 8. A comparison of random VAE samples and their corresponding VAE+GAN samples.

to the inspired music writing found on The Hype Machine and other music blogs, but will spur further research into transfer learning for creative text generation.

However, it is important to always strive to keep social responsibilities and ethical considerations in mind for any machine learning project, even when developing in a seemingly harmless domain. Although one might be tempted to ask what possible harms could result from generating automatic music commentary, there are several ways in which this project is subject to bias, and could potentially be used for harmful applications. For example, if this project contributes to realistic text generation, what does it help to enable? Automated fake Yelp reviews[7], fake news, and troll bots represent some unfortunate possibilities. We must consider how can this research can be re-purposed, and how this should this affect the work that we do. Even if the technique itself is not employed for an objectionable purpose, we know that the commentary generated by the model is heavily biased by the dataset on which it was trained. How can the skew within the data affect which viewpoints are promoted by our trained model? We hope that the discussion we have provided about bias has helped illuminate some of these questions.

[7] https://read.bi/2O3s6Io

6.1. Future Work

Now that @deephypebot has been deployed live on Twitter, it represents an exciting direction for future research. Each ‘like’ and ‘retweet’ given to the bot can provide an important source of training data; namely, it reflects human preferences over the generated text! Learning from human preferences (e.g. (Christiano et al., 2017; Jaques et al., 2018)) is an emergent direction in deep learning research that may be extremely important to questions of AI safety, but is also critical to generating satisfying creative works. By learning what type of samples humans enjoy, we can not only train models that are better equipped to interact with humans in a satisfying way, but we can generate higher quality creative content. In fact, collecting human preference data is really the only way to learn an objective function that reflects the subjective quality of the output of a creative model. Therefore, in future work we intend to use Twitter-provided human preference data to continue to refine our model – perhaps using reinforcement learning – in order to generate better, more compelling music commentary.

Another interesting direction would be to condition the VAE on audio features instead of (or perhaps in addition to) genre. The Spotify API provides features that measure aspects of a song’s mood (danceability, energy, tempo, valence), properties (instrumentalness, loudness, speechiness), and context (acousticness, liveness) – all of which hold interesting potential for informing generated music commentary.

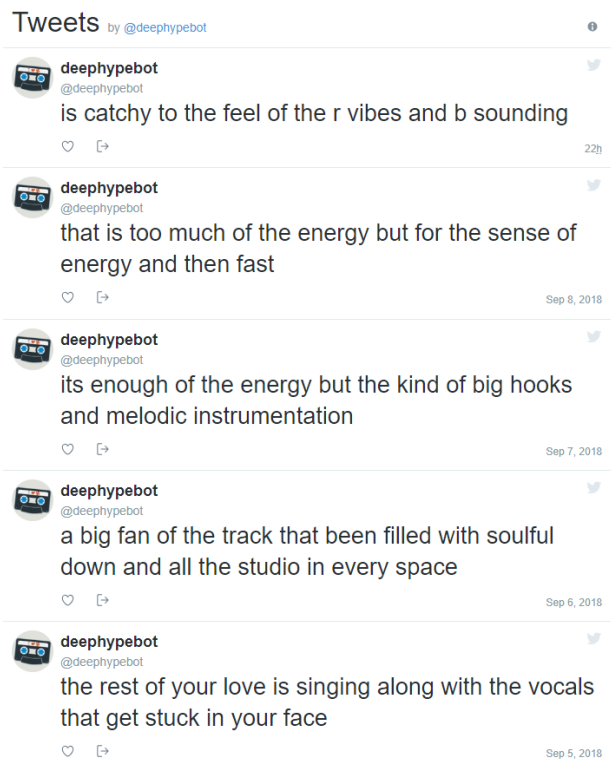

Figure 8. A screenshot of @deephypebot's tweets.

References

Bowman, Samuel R, Vilnis, Luke, Vinyals, Oriol, Dai, Andrew M, Jozefowicz, Rafal, and Bengio, Samy. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Christiano, Paul F, Leike, Jan, Brown, Tom, Martic, Miljan, Legg, Shane, and Amodei, Dario. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

Engel, Jesse, Hoffman, Matthew, and Roberts, Adam. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, 2017.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ha, David and Eck, Douglas. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

Jaques, Natasha, Engel, Jesse, Ha, David, Bertsch, Fred, Picard, Rosalind, and Eck, Douglas. Learning via social awareness: improving sketch representations with facial feedback. *arXiv preprint arXiv:1802.04877*, 2018.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Li, Jiwei, Monroe, Will, Ritter, Alan, Galley, Michel, Gao, Jianfeng, and Jurafsky, Dan. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Li, Jiwei, Monroe, Will, Shi, Tianlin, Jean, Sébastien, Ritter, Alan, and Jurafsky, Dan. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

Roberts, Adam, Engel, Jesse, Raffel, Colin, Hawthorne, Curtis, and Eck, Douglas. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.