Generating Opionated Movie Reviews using GANs

Thomas Nortmann
982524

Lara McDonald
978624

March 26, 2022

Abstract

Generative Adversarial Networks have been immensely successful in generating artificial images almost indistinguishable from real-world examples. This success begs the question, of whether the architecture can also be applied to other domains, such as text. After discussing the challenges of adapting GANs to the textual domain and outlining existing approaches, we implement a LaTextGAN model. The model is trained on the IMDB Reviews Dataset and tasked with generating realistic movie reviews. Additionally, the model takes a sentiment parameter (positive/negative) and should generate opinionated reviews reflecting this sentiment. RESULTS

1 Introduction

Generative Adversarial Networks (GANs) are a well-established tool for the creation of artificially generated images. Having experienced great success in this domain, it is interesting to explore the possibilities of adapting the architecture to other domains, such as text-based tasks.

The basic idea behind the GAN architecture is to let two neural networks play a min-max game against each other, or rather a re-imagined version of the Turing Test. Player One, the Generator, is tasked with creating samples based on a random noise input, while Player Two, the Discriminator, is tasked with deciding whether its input sample belongs to the set of "real" training samples or is simply a "fake" sample created by Generator. The goal of this spiel is for the Generator to be able to generate images that are indistinguishable from "real" samples.

The idea behind our Final Project was to apply GANs, which have been very successful in generating images indistinguishable from actual photographs, to a textual domain. Specifically, we aimed at constructing a model which can generate plausible, opinionated movie reviews. When given a sentiment parameter (positive/negative) the GAN should be able to generate a review that aligns with the sentiment. To achieve this, the basic GAN structure needs to be adjusted to allow for input constructed from this discrete, textual domain, rather than the continuous spaces it was originally intended for. Additionally, vanilla GANs do not accept an external condition for sample generation, such as the sentiment parameter. Thus, a second adaptation to the architecture is necessary for the model to achieve its task.

We chose to implement a variation on LaTextGAN, an architecture proposed by D. Donahue and A. Rumhinsky in 2019 [3]. LaTextGAN circumvents the problem of having discrete outputs by utilizing a Variational Autoencoder to encode the input samples before GAN training. We also adapted the LaTextGAN architecture to allow for the additional sentiment parameter.

RESULTS

This paper aims to document our research and development process. We begin by further detailing the task at hand and the dataset we are working with. Afterward, we will shortly review relevant literature and architectures, discussing their advantages and flaws regarding our chosen task. We will then describe our choice of architecture and will lastly review and discuss our results.

2 The Dataset

The model is trained on the *Large Movie Review* Dataset [5], also known as the *IMDB Reviews* Dataset, which is publicly available in TensorFlow datasets. The dataset contains 100,000 user-generated movie reviews from the online movie database IMDB. Half of the reviews are unlabeled, while the other 50,000 are labeled accord-

ing to the primary sentiment of the review, e.g. positive or negative. The sentiment classes were assigned according to the mandatory rating an IMDB user needs to leave alongside their review. Movies are rated on a 10-star scale, with 1 star equating to strong discontent and 10 stars to strong satisfaction.

The IMDB Reviews Dataset is advantageous in multiple ways for this project. Firstly, the labeled examples are balanced, i.e. there are 25,000 reviews classified as being positive and an equal amount of examples classified as negative. Since our GAN aims to generate reviews of both sentiments, an unbalanced dataset could skew results. Furthermore, the dataset only contains reviews with strong sentiment polarity. To be included in the negative sentiment class, a review's rating must be less or equal to 4 stars, while only reviews with a rating of 7 stars or higher are included in the positive sentiment class. The model can thus learn to generate sentences with a clear sentiment.

2.1 Preprocessing Steps

Each sample in the dataset, as it is available in TensorFlow, is a *FeaturesDict* containing a *label* and a *text*. The *label* is an integer of value 0 (negative sentiment) or 1 (positive sentiment), while the *text* is the corresponding movie review in plain text. To use the data samples as input for our model, several preprocessing steps need to be performed on the plain text component of the tensors.

First, the text is cleaned by removing special characters. The resulting text is now tokenized with the BertTokenizer from TensorFlow Text. To each tokenized sentence, a [START] and [END] token is added to the beginning and end of the sentence, respectively. This is a necessary step, as it enables the Generator to learn how to begin a review and more crucially when to end it.

3 Background

3.1 Generative Adversarial Networks

Basic GAN Architecture

The first GAN architecture was proposed in 2014, in the paper "Generative Adversarial Nets" by Goodfellow, et al. Its architecture fulfills the basic principle of the min-max game, as described above. Two neural networks, the Generator, and the Discriminator work against one another: the Generator aims to generate samples resembling real-world samples closely enough to fool the Discriminator, who tries to distinguish between the fake, generated samples and the real-world data. The underlying principle of this architecture is that the Generator approximates a transformation function, which maps a random input distribution to the distribution underlying real data samples. On the other hand, the Discriminator

approximates the distance between the generated sample and the real underlying distribution. [4] [11]

For a GAN to converge, it is necessary to reach Nash equilibrium between the Generator and Discriminator. In game theory, the Nash equilibrium describes the state where both players' strategies are optimal, considering the other player's strategy. However, GANs are typically optimized using Stochastic Gradient Descent. As SGD isn't designed to find this equilibrium between Generator and Discriminator, basic GANs can suffer from failure to converge [8]. If the Discriminator outperforms the Generator by too much, vanishing gradients may occur, impeding convergence. If the roles are reversed and the Generator consistently fools the Discriminator, training may continue after the point of optimality, worsening results. Another common issue with GANs is mode collapse, which occurs when the Generator generates a successful example and cannot deviate from this, thus only generating the same example repeatedly.

Wasserstein GANs [1]

There are various changes to the standard GAN architecture that address its problems with slow convergence, lack thereof, and mode collapse. A widely accepted improvement is the use of Wasserstein Distance ("*Earth Mover's Distance*"), instead of the Jensen-Shannon Divergence used in a vanilla GAN, as a loss function. In a standard GAN, the JS Divergence measures the distance between the real sample distribution and that of the fake samples generated by the Generator. During Discriminator training, this distance should then be minimized. In WGANs the Discriminator is transformed to what the authors call *The Critic*. Instead of discriminating between real and fake samples, the Critic merely scores its input sample on how real it is.

The advantage of WGANs lies in the continuously differentiable Earth Mover's Distance, which allows for the Critic to be trained to optimality without risking vanishing gradients, as is the case in the vanilla architecture. As the antagonist to the Generator is defined differently, there is no need to reach Nash equilibrium, thus stabilizing training. The issue of mode collapse is alleviated, as well.

Conditional GANs [6]

After GANs proved to be an effective tool for artificial image generation, the desire to directly modify the image class arose. Conditional GANs allow for this modification by training both Discriminator and Generator on an additional input parameter. This parameter is simply concatenated with the input sample to either component while training occurs as usual.

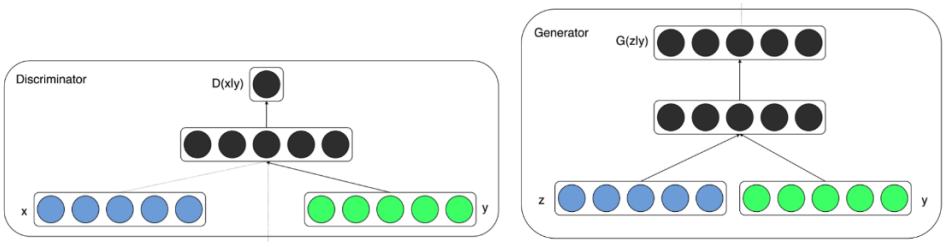

Figure 1: The Conditional GAN Generator and Discriminator Components. Where x denotes the training sample, y is the additional parameter, and z is a random noise input

3.2 Variational Autoencoders

The Variational Autoencoder is an adaptation of the basic Autoencoder, allowing for more advanced sample generation. The architecture of autoencoders consists of an encoder component and a decoder component, both neural networks. The goal of the autoencoder during training is for its output to correspond to its input, while not simply learning the identity function. To achieve this, the data is passed through the encoder to a bottleneck, the embedding layer, only for this to be transformed to the original input by the decoder. At the embedding layer, the input data is now compressed while still capturing the essential characteristics of the input.

Variational Autoencoders build upon this architecture by additionally learning the underlying distribution of the embeddings. Using the reparameterization trick, i.e. assuming an underlying Gaussian distribution, noisy samples can now be generated from the embedding, allowing for variations on the original inputs. [7]

4 Related Approaches

When applying the GAN architecture to the textual domain, the main hurdle to overcome is how to handle the value space of text-based data [10]. Compared to the continuous image space GANs were originally intended for, textual symbols are discrete and thus non-differentiable. This poses a major problem during training, as gradient backpropagation is no longer meaningfully possible [2]. There are multiple proposed workarounds to this issue, such as pre-training the model with Maximum Likelihood Estimation [9]. Other approaches are to make use of Reinforcement Learning Strategies or Word Embeddings.

4.1 SeqGAN [12]

A well-known example of a GAN architecture with Reinforcement Learning Properties is the SeqGAN model proposed in 2017. The generation problem in SeqGAN is posed as a reinforcement problem, with the generator aiming to maximize the expected reward of a generated sentence given by the discriminator. The more plausible the sentence, the higher the reward. However, the generator does not produce a full sequence immediately. Instead, it samples the next token and then enacts a Monte Carlo Search over the rolled-out policy gradient to calculate the expected reward of a sequence. [11]

4.2 GAN2Vec [2]

A different solution to the task is the GAN2Vec architecture, proposed in 2019. The objective of the generator is not to create sentences in a human-readable discrete format, but rather to generate Word2Vec embeddings. This overcomes the hurdle of discrete spaces, as the resulting vectors are no longer one-hot vectors but continuous embedding vectors. The paper also proposes a modification to GAN2Vec which allows for an additional conditional parameter to be given to the model, such that it is possible to generate text accordingly.

5 Model

To achieve this specific task utilizing GANs, a few considerations regarding the model's architecture need to be made. REPHRASE: Primarily, the architecture must eliminate the problem of using a model made for continuous spaces on the textual domain, as is the case when generating movie reviews. In addition, the model must be able to generate reviews under the intended sentiment, i.e. whether the review should be positive or negative. To overcome the first concern, we decided to focus on the usage of word embeddings, rather than reinforcement learning. Regarding the sentiment parameter, there was the option to implement GAN2Vec's conditional variation or to modify a different architecture to comply with the idea of Conditional GANs. We settled on the latter, as it allowed us to partially implement an established architecture, while also being able to extend the existing approach to fit the task at hand.

5.1 LaTextGAN [3]

The LaTextGAN architecture was proposed in 2019 to adapt GANs to discrete spaces without the need to use Reinforcement Learning Techniques. Unlike vanilla GANs, LaTextGAN consists of a GAN component and an additional Variational

Autoencoder, which in itself is further comprised of two sub-components: an Encoder and a Decoder.

The Generator and Discriminator components in LATextGAN are realized using LSTM cells in a ResNet architecture. It is important to note that the GAN component uses Earth Mover Distance and thus falls in the class of Wasserstein GANs. The Variational Autoencoder also utilizes LSTM cells to process the input sample at each time step and to model the long-term dependencies between words. At the embedding layer (colored orange in Figure 2) of the VAE, the sample is encoded as a low-dimensional *real* vector in the latent space. This representation is the key to circumventing the problem of using GANs on discrete domains and enables backpropagation.

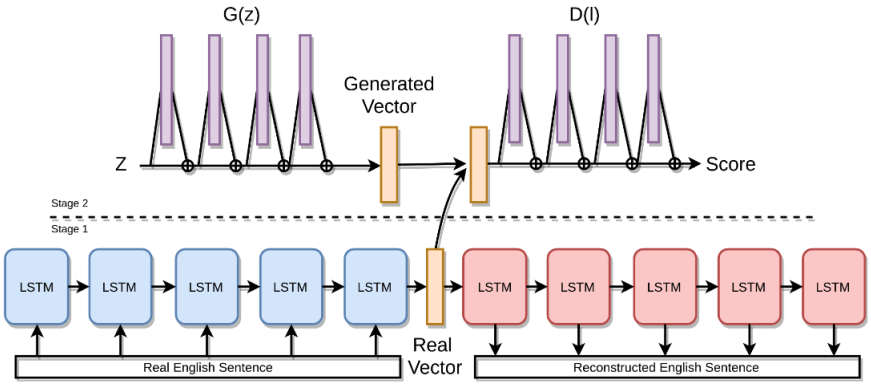

Figure 2: The basic LaTextGAN architecture

Due to the additional VAE component, training has to occur in two distinct stages. Stage 1 focuses solely on the VAE, while the GAN is fixed. During Stage 2, the trained encoder and decoder are used to transform the discrete textual representation into real vectors and vice versa. The Generator no longer is tasked with generating novel sentences, but rather real-valued latent representations. To accommodate this modality change, the Discriminator is trained on the latent encoding of the training samples, to score the plausibility of the generated encoding to be a real sentence. The decoder can later be used to transform the latent vector back into natural language.

5.2 Final Architecture

Underlying our generative model is the LaTextGAN architecture, however, adapted to fit the specific task definition, i.e. generating plausible, opinionated movie reviews, conditioned on the given sentiment parameter (positive/negative).

Variational Autoencoder

Generative Adversarial Network

6 Results

6.1 Evaluation Scores

7 Discussion

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[2] Akshay Budhkar, Krishnapriya Vishnubhotla, Safwan Hossain, and Frank Rudzicz. Generative adversarial networks for text using word2vec intermediaries. *CoRR*, abs/1904.02293, 2019.

[3] David Donahue and Anna Rumshisky. Adversarial text generation without reinforcement learning. *arXiv preprint arXiv:1810.06640*, 2018.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[6] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[7] Achraf Oussidi and Azeddine Elhassouny. Deep generative models: Survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2018.

[8] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans.

[9] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China, July 2015. Association for Computational Linguistics.

[10] Guy Tevet, Gavriel Habib, Vered Shwartz, and Jonathan Berant. Evaluating text gans as language models. *arXiv preprint arXiv:1810.12686*, 2018.

[11] Zesen Wang. Generative adversarial networks in text generation. Master's thesis, 2019.

[12] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient, 2016.