Text Generation using Generative Adversarial Networks

A Project Report Submitted in the
Partial Fulfillment of the Requirements
for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

Golla MachiliKanth Yadav	**18881A1223**
Mohammed Riyan Ali	**18881A1240**
Sushma Lagusani	**18881A1256**

SUPERVISOR

Dr. M. Ramachandro

Associate Professor,Information Technology

Department of Information Technology

VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified

Kacharam, Shamshabad, Hyderabad - 501218, Telangana, India

May, 2022

VARDHAMAN COLLEGE OF ENGINEERING
(AUTONOMOUS)
Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

Department of Information Technology

CERTIFICATE

This is to certify that the project titled **Text Generation using Generative Adversarial Networks** is carried out by

Golla MachiliKanth Yadav	**18881A1223**
Mohammed Riyan Ali	**18881A1240**
Sushma Lagusani	**18881A1256**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology** during the year 2021-22.

Signature of the Supervisor
Dr. M. Ramachandro
Associate Professor

Signature of the HOD
Dr. Muni Sekhar Velpuru
Associate Professor and Head,IT

Project Viva-Voce held on _____

Examiner

Acknowledgement

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr. M. Ramachandro** ,Associate Professor and Project Supervisor, Department of Information Technology, Vardhaman College of Engineering, for his able guidance and useful suggestions, which helped us in completing the project in time.

We express our heartfelt thanks to **Dr. M. Ramachandro**, Associate Professor and Project Coordinator, for his suggestions invaluable inputs and assessment really helped us to shape this report to perfection.

We are particularly thankful to **Dr. Muni Sekhar Velpuru**, the Head of the Department, Department of Information Technology, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R. Ravindra**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartful thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Information Technology department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

Golla MachiliKanth Yadav

Mohammed Riyan Ali

Sushma Lagusani

Declaration

We hereby declare that the work described in this report entitled **Text Generation using Generative Adversarial Networks** which is being submitted by us in partial fulfillment for the award of **BACHELOR OF TECHNOLOGY** in the Department of Information Technology, Vardhaman College of Engineering to the Jawaharlal Nehru Technological University Hyderabad.

The work is original and has not been submitted for any Degree or Diploma of this or any other university

Signature of Student

Golla MachiliKanth Yadav	18881A1223
Mohammed Riyan	18881A1240
Sushma Lagusani	18881A1256

Abstract

Traditionally GANs (Generative Adversarial Networks) are used for image generation but here we are generation text using GANs. They have two subnetworks models. A generator model that trains to generate new examples and a discriminator model that attempts to generate deepfake words or, as a result, produces deepfake words.Both models involve the technique of GAN and are trained on large scale dataset. The dataset we are going to use in this project is Animal dataset which consists of animal related data. In this we build LSTM (Long-Short Term Memory) as generator and also discriminator as well. GANs based text generation has previously not been explored. LSTM and RNN(Reccurent Neural Network) are used to analyse and extract information from sequential data. For accurate text generation, we used the LSTM neural network model. We built two LSTM networks in this project: a generative network and a discriminative network. Simply put, the generative network generates text. The discriminative network has been taught to distinguish between generated and handwritten text. The goal is to train the generative network to increase the likelihood that the discriminative network believes the generated text is human. This is similar to Goodfellow , 2014's Generative Adversarial Network (GAN).

Keywords: Generator, Discriminator, Long Short Term Memory(LSTM),deep fake,Reccurent Neural Network

Table of Contents

List of Figures

Abbreviations

Abbreviation	Description
GAN	Generative Adversarial Network
NN	Neural Network
DCGAN	Deeply Convoluted Generative Adversarial Network
AI	Artificial Intelligence
RGB	Red Blue Green
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network

CHAPTER 1
INTRODUCTION

CHAPTER 1

Introduction

1.1 Introduction

In 2014, Ian Goodfellow published the task of using two separate neural networks to generate synthetic data with properties similar to real data. This work has stimulated the research community's interest in generating realistic images, videos, and general synthetic structure data.

GAN is an unsupervised deep learning method. This is typically implemented using two neural networks, a generator and a discriminator. These two models compete with each other in the form of game settings. The GAN model is trained based on the actual data and the data generated by the generator. The role of the classifier is to distinguish between fake and real data. Because the generator is a training model, it may initially generate weak or completely noisy data that does not reflect the actual distribution or properties of the actual data. The main purpose of the generator model is to create artificial data that can successfully pass through the discriminator. The model gets noise (usually Gaussian noise) and produces fake data. The generator needs to learn in order for the discriminator to win the positive classification. If one of these generated images is successfully recognized as "false", the loss of the generation step is calculated. [1] Discriminators need to learn to gradually identify these fake images. If the model does not detect false data, the discriminator receives a negative loss. An important concept is to train the generator and discriminator at the same time.

The Generative Adversarial Network (GAN) is a machine learning (ML) model in which two neural networks compete with each other to make more accurate predictions. GANs are typically run unattended and use a collaborative zero-sum game framework for learning.[2] The purpose of the generator is to synthesize output that can easily be confused with the actual data. The

purpose of the discriminator is to identify the artificially generated output received.

- A generated network that takes random input z and returns a G (z) that must follow the desired probability distribution.
- An identification network that receives words and classifies whether the generated words are genuine or fake.

The development of the Generative Adversarial Network (GAN), which can generate high-quality, high-resolution samples from images, includes key applications such as image repair, 3D data, domain conversion, video compositing, image manipulation, semantic segmentation, and semi-supervised learning.

1.2 Problem Definition

Traditionally, GAN (Generative Adversarial Networks) was used for image generation, but here GAN is used to generate text. They have two subnetworks models. A generator model that trains to generate new examples and a discriminator model that attempts to generate or generate deeply forged words according to the dataset used, called animal datasets. Both models include GAN technology and are trained on large datasets. GAN-based text generation has not been studied so far. The LSTM can be used as both a classification model and a generative model, but here we use the LSTM as a generator and as a discriminator that provides the application with deeply forged words. The purpose of the discriminator is to identify the artificial output obtained. The goal is to train the generation network to increase the likelihood that the identification network will believe that the generated text is human.

This project is useful for both businesses and the general public.

1.3 Objectives

- The main objective is to generate words that can be used for security purposes like OTP generation.

- Our model generates the deep fake words according to the dataset
- Mainly GANs are used for image generation but in this project we are using GANs for Text Gen which is not explored. [3]

1.4 Limitations of Project

GAN for text

- GAN is not directly applicable for text generation so we are using LSTMs.

Evaluation

- One of the key points is the approach to the quality of the generated data.
- The discriminator helps to verify that the generated data is correct.

Loss Function

- The GAN model has the unusual property of simultaneously training the generator and discriminator.
- This necessitates a loss function that balances discriminator and counter-discriminator training (generator).

Determination of Convergence

- The game's ability to find the perfect champion is hampered by competition between discriminators and generators. Slow convergence is a known issue with GAN models.
- Both models have been meticulously designed to maximise revenues while minimising losses.
- You must strike a balance between training time and product quality. Several factors influence how quickly or slowly the formation occurs. [4]

CHAPTER 1
INTRODUCTION

CHAPTER 2

Literature Survey

2.1 Existing System

Many researchers have been working on "Twitter sentiment analysis" in recent years. Early on, the idea was to create a binary categorization system that exclusively assigned positive or negative bipolar classes to opinions or reviews. [5]

Pak and Paroubek (2016) suggested a model for objectively identifying tweets as favourable or negative. They created a Twitter corpus by using the Twitter API to collect tweets and automatically annotate them with emoji. We constructed an emotion classifier based on the multinomial naive Bayesian technique with characteristics like Ngram and POStags using this dataset. Because the training set only included tweets with emoji, it was less effective.[6]

Parikh and Movassate (2015) used the naive Bayes biggram model and the maximum entropy model to classify tweets. The naive Bayes classifier outperforms the maximum entropy model, according to the researchers. Go and L. Huang (2015) proposed a remote monitoring approach for sentiment analysis of Twitter data. The training data was made up of tweets with emoji that served as noisy labels. Their models are built using Naive Bayes, MaxEnt, and Support Vector Machines (SVMs). Unigrams, bigrams, and POS were their functional regions. They conclude that SVM outperforms other models and Unigram outperforms other features.[7]

Barbosa and colleagues (2017) For classifying tweets, we created a two-step automatic sentiment analysis algorithm. They classified tweets as objective or subjective, and then classified subjective tweets as positive or negative in the second step. Retweets, hashtags, links, punctuation marks, and exclamation marks were all employed in the functional space, along with features like prior

word polarity and POS. [8]

The Firehouse API gave Twitter streaming data to Bifet and Frank (2016). It publishes all of the messages from each user in real time. They used the multinomial naive Bayes method, stochastic gradient descent, and the Hoeffding tree to test their theories. They found that when utilised at a suitable learning rate, the SGD-based model outperformed the other models. [9]

Agarwaletal is a kind of agarwaletal (2018) We created a three-part model to categorise mood into good, negative, and neutral categories. They used models including unigram models, feature-based models, and tree-core-based models to test their theories. Tweets are represented as a tree in the tree core-based paradigm. The Unigram model has over 10,000 features, while the feature-based model has only 100. They find that the capacity to combine the previous polarity of the word with the part-speech (pos) tag is critical and plays a key role in the categorization task. The model based on tree cores outperformed the other two. [10]

Davidov et al. (2019) proposed a method for classifying emotions using Twitter custom hashtags in tweets. Use different feature types including punctuation marks, single words, ngrams, and patterns to produce a single feature vector. You can mix and match mood categories. They gave emotion labels by creating feature vectors for each sample in the training and test sets using the KNearest Neighbor technique.[11]

PoWei Liang et al. (2017) collected Twitter data using the Twitter API. Training data is divided into three groups (camera, movie, mobile). The data is divided into three categories: good, negative, and no opinion. Opinionated tweets that have been filtered. [3]The Unigram Naive Bayes model was implemented, with the Naive Bayes independence requirement simplified. To reduce undesired characteristics, we also used mutual information and a chi-square characteristic extraction method. Finally, the tweet's direction is predicted. That is, whether it's positive or negative. [12]

Pablo and his colleagues For recognising the polarity of English tweets, we introduced a variant of the naive Bayes classifier. The Naive Bayes classifier has been created in two separate forms. That is, baseline (taught to classify

tweets as positive, negative, or neutral) and binary (trained to classify tweets as positive, negative, or neutral) (classify as positive and negative using polar lexicons; neutral tweets are ignored). Headwords (nouns, verbs, adjectives, and adverbs), polar lexicons and multiwords from multiple sources, and valence shifters were among the functions considered by the classifier. [13]

Turny, for example. The bagofwords approach of sentiment analysis was applied. It presented the document as if it were merely a collection of words, with no regard for the links between them. The emotions of each word were identified, and these values were blended using numerous aggregation processes to determine the emotions of the entire document.[14]

Kamp and colleagues The emotional content of words was determined using the WordNet lexical database on many dimensions. They used WordNet to build a distance metric and estimate the semantic polarity of adjectives. [15]

Xia et al. combined multiple feature sets and classification algorithms to create an ensemble framework for emotion classification. We employed two feature sets (the link between voice information and words) and three fundamental classifiers in their research (naive bayes, maximum entropy, support vector machine). To improve accuracy, they used an ensemble approach to emotion categorization, which included fixed combinations, weighted combinations, and meta-classifier combinations.[16]

Al. Luoet He focused on the difficulties and effective approaches for obtaining opinions from Twitter tweets. Getting an opinion on Twitter might be difficult due to spam and different languages.[17]

2.2 Limitations of Existing Systems

However, computer programs struggle to recognize sarcasm, sarcasm, denial, jokes, exaggerations, and anything else that is easily identifiable by humans. And if these are not recognized, the results can be distorted. "Disappointed" can be classified as a negative word for sentiment analysis purposes, but it should be classified as positive in the sentence "I was not disappointed".[18] You can easily say, "I love the hotel's huge pool!" If this statement is accompanied by a photo of a small pool, it is considered ironic. Automated

sentiment analysis tools, on the other hand, probably won't and will probably not classify it as an example of positive emotions.

Reliable sentiment analysis contexts may not be sufficient, especially for short sentences and texts such as those found on Twitter and Facebook. However, in general, Twitter has a reputation for being a great source of sentiment analysis, and with the new word count in tweets, Twitter could be even more useful. Therefore, automated sentiment analysis tools are very good at analyzing textual opinions and attitudes, but they are not perfect. If you use a tool like typely to analyze the text and see if it conveys the desired mood to your readers and audience, you can combine the results with human judgment to make it difficult for the tool to make a decision.

2.3 Proposed method and Advantages

Here in our project we used GANs for text generation. Traditionally, GAN was used for image generation, but here GAN is used to generate text. Text generation based on GAN has not been investigated so far. Here, the LSTM is used as the generator and the LSTM is used as the discriminator. Since it uses an animal dataset, it consists of animal name classifications.[19]In which we have this many-to- many relationship. The input as a dataset given to discriminator as words, output is many generated deep fake words.

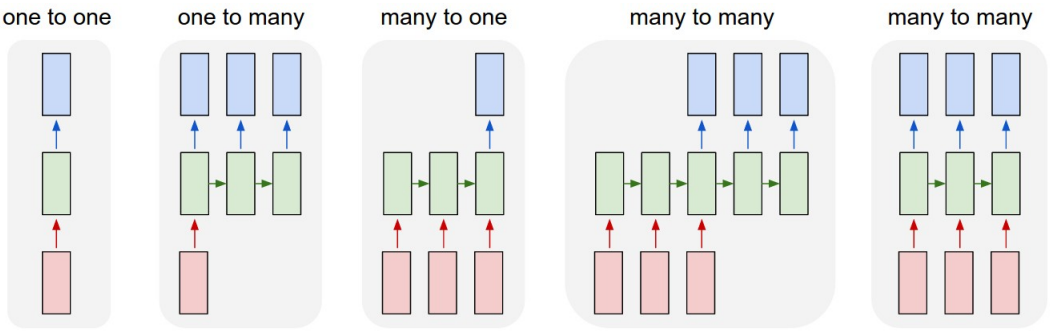


Figure 2.1: Many-to-Many relationship

Advantages of proposed methodology

The advantages of utilising LSTM as a generator Long Short Term Memory is a recurrent neural network that, when compared to RNNs, is better at

modelling time-series assertions and long-range relationships. An improved version of RNNs are only kept in memory for a brief period of time.[20]

- The accuracy of predictions is improved by using LSTMs. Backpropagation of non-damped errors.
- LSTMs can manage noise and continuous readings for long time delay concerns.
- There are no changes to the parameters. Over a lengthy period of time, there is a lot of recollection.
- LSTMs required more parameters than CNNs, but only about half of DNNs.
- Long Short Term Memory (LSTM) is a technique for predicting data sequences.
- The most time-consuming is training, but it has the advantage of letting you to examine large sequences of inputs without expanding the network.

CHAPTER 1
INTRODUCTION

CHAPTER 3

Analysis

3.1 Introduction

GAN has solved many problems with generative models and has influenced other AI techniques, but it still has its limitations. GAN has adopted the idea of hostile learning, but the convergence of the model and the existence of equilibrium points have not yet been proven. The training process requires ensuring the balance and synchronization of the two opposing networks. Otherwise, it will be difficult to get good training results. However, controlling the synchronization of two opposing networks can be difficult and can lead to instability in the training process. In addition, as a generative model based on neural networks, GAN has some common flaws (that is, poorly interpretable) in neural networks.[21]

3.2 Software Requirement Specification

3.2.1 Software Requirements

- Programming Language : Python3
- Framework Used : Tensorflow
- Development Platform : Google Collaboratory
- Training Time : Few Minutes for training words from animal dataset.

3.2.2 Hardware Requirements

- Processor : Intel(R) Core(TM) i5-8265U
- CPU : 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s)
- Ram : 8GB

3.3 Flowchart

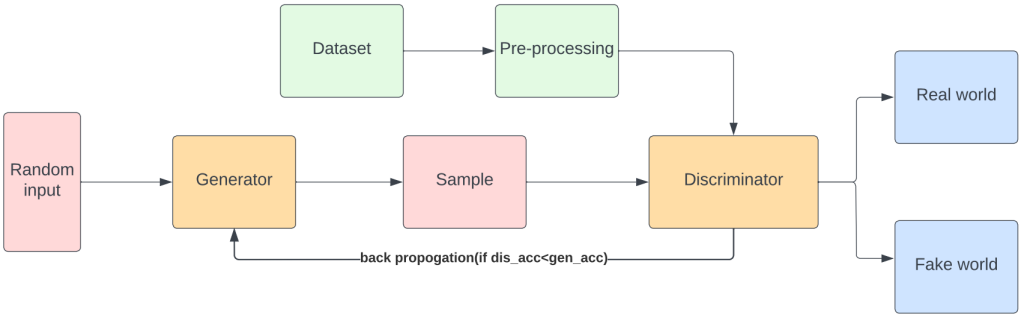


Figure 3.1: Flow Chart

STEPS

- Firstly train discriminator on animal text document which consists of animal names as words.
- Nextly freeze discriminator training and start giving random seed input to generator.
- Generator generates fake words based on given random seed. Those generated words are given to discriminator which distinguishes as actual or copy.
- If the discriminator's accuracy is more only then back propogation takes place. Back propogation takes place until generator accuracy is more and discriminator accuracy is less.
- When the above condition is satisfied then generator outputs words after each epoch.

GANs, or Generative Adversarial Networks, are a type of generative modelling that employs deep learning techniques such as convolutional neural networks.[22]

Generated modelling is an unsupervised machine learning activity that finds and learns the regularity or patterns in incoming data autonomously. This enables you to utilise the model to create or output additional examples based on the original dataset.

GANs are a clever way of schooling a generative version by framing the problem as a supervised learning problem with two sub-fashions: the generator version, which we train to generate new examples, and the discriminator version, which attempts to categorise examples as either real (from the domain) or fake (from the domain) (generated). The fashions compete in a zero-sum competition until the discriminator version is deceived around half of the time, indicating that the generator version is creating plausible examples.

GAN is a fascinating and quickly evolving field that harnesses the power of generative models to generate realistic instances of a variety of problems.[23]

CHAPTER 1
INTRODUCTION

CHAPTER 4

Design

4.1 Introduction

In our model we are designing LSTM as generator and LSTM as discriminator.The Long Short Term Memory Network can recognise long-term dependencies. They are mostly used to solve timeseries data and can learn patterns from prior inputs and predict the future. Data is used to train features in Neural Networks. LSTM performs the same thing. [24] LSTM layer design might be challenging. You could get stuck deciding on the proper number of layers, size, and other factors when designing for the first time. The model may or may not be a good match. A good model is usually the result of a lot of trial and error.

How do we design a model?
Choose the one that functions similarly to the model and utilise it. While this method works in many situations, it is not appropriate in all. This is especially true if you're working on a novel problem or conducting groundbreaking research. While designing your own model, you can take inspiration from popular models.[25]

How do we decide how many layers and units to use?
Most of the time, it's easier to start with a smaller model and gradually raise the size. The number of layers and units will rise as the model size is increased. In the model sense, it is usually better to reduce rather than raise a very vast and thin network because optimising it is quite tough.

How to choose the activation function , batch size and epochs ?
Except for the output layer, it could be easier to select Relu. For neural

networks, a stack size of 32 is sufficient, but if the gradient is too great, you will need to employ a huge number of stacks. Epoch refers to the entire data set that a neural network is passing back and forth. You can choose from a huge variety of epochs when choosing one.[26] ¿¿ 100, with an early stop. To stop the model and avoid overfitting, utilise early stopping. Before the model is changed during the era, an early stop is applied.

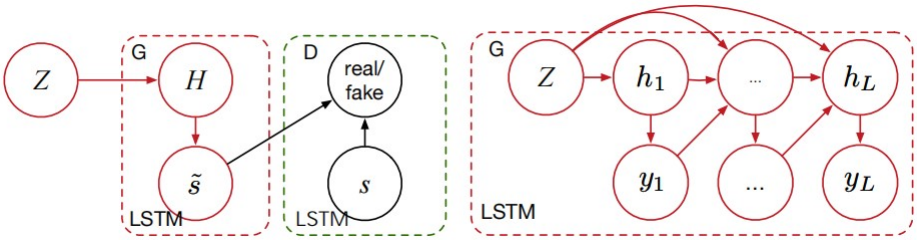


Figure 4.1: Design

Here we are making LSTM as both generator and discriminator. This provides a flexibility of two different architectures under single model.So that our accuracy gets increased As you can see in the flowchart in Figure , when a random input is given to the generator (in this case the LSTM) and the sample output and this output are generated, the design is effectively displayed and initially easy to understand. It is passed as an input to the discriminator. The discriminator takes two inputs from the generator output, similar to a real text sample, and then propagates back to the generator while the generator tries to show that the output is untrue.

GAN is a method for unsupervised deep learning. Two neural networks, a generator, and a discriminator are commonly used to implement it. [27] In the shape of a gaming environment, these two models compete with one another. The GAN model is trained using both real-world and generator-generated data. The discriminator's job is to tell the difference between bogus and true data. Because the generator is a training model, it may generate data that is weak or fully noisy at first, which does not match the actual data's distribution or features. The generator model's principal goal is to create artificial data that can pass through the discriminator successfully.

The generator's objective is to create output that can be readily fooled with real data. The discriminator's job is to recognise the falsely created

output that has been received.

4.2 Module Design and Organization

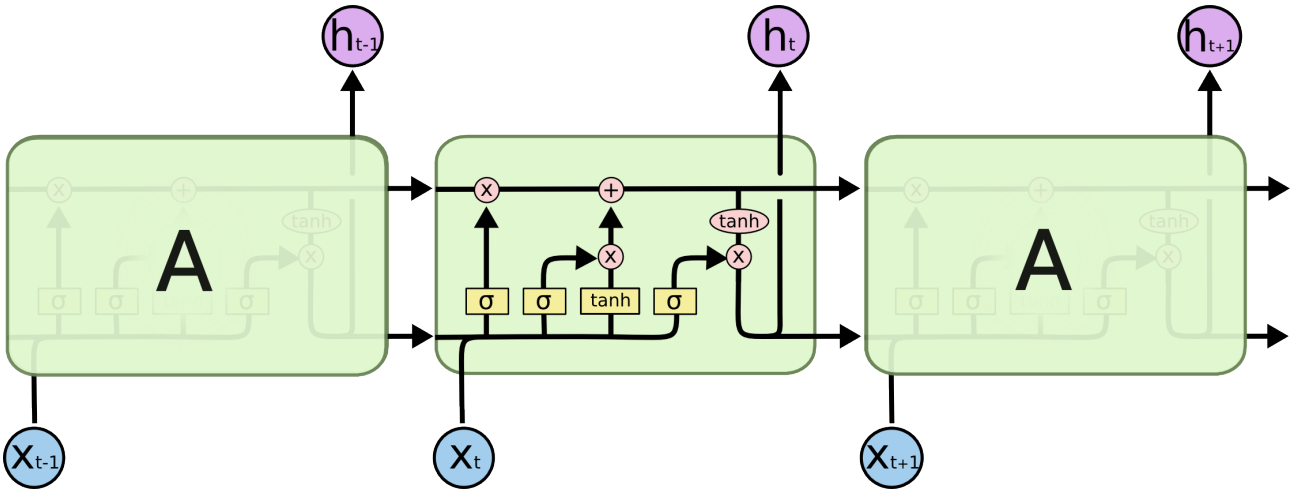


Figure 4.2: An LSTM's repeating module has four levels that interact.

Each row in the diagram above transmits the entire vector from one node's output to the other node's input. Pointwise operations are shown by pink circles, such as: B. Add vector, and the trained neural network layer is shown in the yellow box. When rows are merged, concatenation occurs, and forks occur when the contents of a row are duplicated and a copy transferred to another location. The LSTM relies on the state of the cell, which is represented by the horizontal line that goes through the top of the chart. The condition of the cell is comparable to that of the assembly line. The entire chain is made up of only a few modest linear interactions. It's quite simple to relocate without losing data.A gate, which is carefully regulated by an LSTM, can remove or add cell state information. A gate is a mechanism for passing information selectively. A sigmoid neural network layer and pointwise multiplication are included. It produces an integer in the range 0 to 1 that indicates the amount that each component is allowed to pass through. "Don't pass anything!" If "all pass", the value means zero. Means a value of 1. Three of these gates are present in the LSTM to protect and control the state of the cell. The first step in LSTMs is to determine which cell state information to discard. The sigmoid layer, the "gate oblivion layer," makes

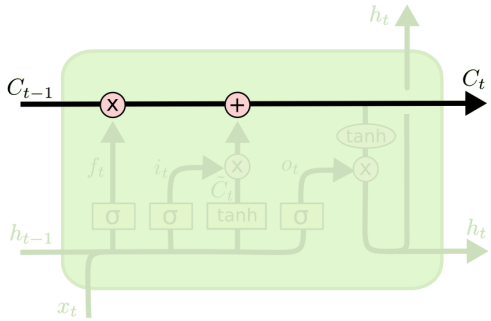
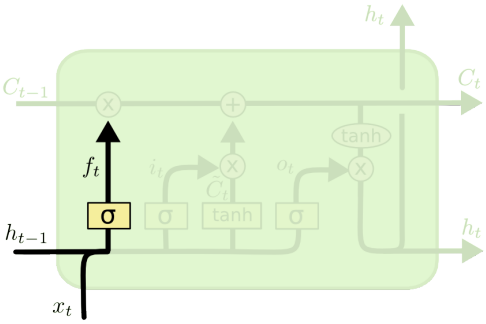


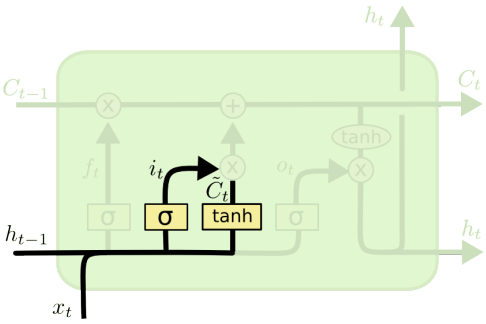
Figure 4.3: Cell-state



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 4.4: Forget-gate layer

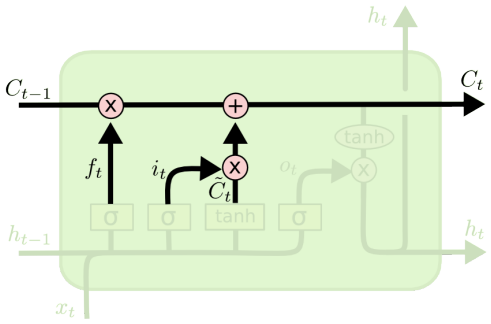
this decision. Consider a language model that uses the previous word to predict the next word. In this situation, the cell status can indicate the current subject's gender, and the appropriate pronoun can be used. When viewing a new subject, I try to forget about the prior subject's gender.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 4.5: Input gate layer

The next step is to determine what new information will be stored in the cell state. This is made up of two parts. [28] A sigmoid layer called the "input gate layer" choose which variables to update first. The Ct vector of new candidate values that could be added to the state is then generated by a tanh layer. In the next phase, we'll combine these two to create a state update. We'd like to change the gender of the old subject in the cell state



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 4.6: Output-gate layer

to that of the new subject for our language model.

Finally, you must choose what to create. This output is filtered and dependent on cell state. To detect which side of the cell state is being output, first run the sigmoid layer. The cell state is then passed through tanh (to force the value to be between 1 and 1), the sigmoid gate output is multiplied, and just the selected part is output. Having just observed the subject, the language model may want to output information related to the verb the next time the verb comes. LSTMs have been a big step forward in what RNNs can do. It's natural to wonder if there is another big step forward. "Yes!" Said the majority of researchers. There are next levels and it's all about paying attention! The goal is to select the information to investigate from a larger data pool at each stage of the RNN. When using RNNs to create captions, in this case you can select different parts of the image to display for each generated word. In fact, Xuetal. (2015) Do exactly that. If you want to learn more about attention, it may be a good place to start.

CHAPTER 1
INTRODUCTION

CHAPTER 5

Implementation

5.1 Introduction

In this project, we will implement GAN for TEXT GEN. Traditionally, GAN's was used for photo generation, but here GAN is used to generate text. Text generation based on GAN has not been investigated so far. Here, the LSTM is used as the generator and the LSTM is used as the discriminator. Because we are using an animal dataset, it consists of the names of the animals in this many-to-many relationship. The input is a string of words and the output is also a string of words.GANs have lately been used for classification tasks, and they frequently use the same design for classification and identification. This, however, necessitates the model converging to a separate data distribution for each task, which can impair overall performance. [29]GAN can be used for text. However, there is a problem with the combination of how GAN works and how neural networks usually generate text. GAN works by propagating the gradient through a generator and discriminator configuration. The text is usually produced by placing the last softmax layer on the token space. H. The output of the network is usually the probability of generating each token (that is, a separate stochastic unit).

5.2 Technology

The Generative Adversarial Network (GAN for short) is a deep learning-based approach to generative modelling. The Generative Adversarial Network (GAN) is a machine learning (ML) model in which two neural networks compete for accuracy in prediction. GANs are typically unattended and learn through a collaborative zero-sum game structure.

The generator and discriminator refer to the two neural networks that make

up a GAN. A convolutional neural network serves as the generator, while a deconvolutional neural network serves as the discriminator. The generator's objective is to provide outputs that may be mistaken for real data. The discriminator's purpose is to determine whether of the outputs it receives were generated intentionally. GANs, in essence, generate their own training data. The generator begins to create higher quality output as the feedback loop between the opposing networks continues, and the discriminator becomes better at categorising the artificially generated data. Generated modelling is an unsupervised machine learning activity that finds and learns the regularity or patterns in incoming data autonomously.This enables you to utilise the model to create or output additional examples from the original dataset. To frame the task as a supervised learning problem, GAN uses two submodels: a generator model that trains to generate new instances and a discriminator model that tries to evaluate the examples as real examples. It's a clever technique to train your generative model this way. fake (from the domain) (generated). In a hostile zero-sum game, the two models will be trained jointly until the discriminator model is deceived around half of the time. The generator model, in other words, generates a credible example.

5.3 Method of Implementation

For text creation, we employ GANs in our project. GANs has traditionally been used to generate images, but here it is being utilised to generate text. Text generation with GAN has not yet been examined. Generators and discriminators are two types of subnetworks. It combines the flexibility of a single model with the functionality of two distinct systems. LSTMs are used as discriminators and generators. This one-to-one relationship exists. The input is a set of words, and the created words are the result.[30]

In this task, we will introduce two groups of options, generator and discriminator.

LSTM as generator:

The first set of options is a Recurrent Neural Network with a peephole LSTM cell as the base unit. Joint affine transformations are applied to them to

transform the output of the RNN into a probability distribution. Equipped with an LSTM unit, the RNN also acts as a discriminator. The state of the last unit performs a regression and provides a score that indicates whether the sample is from real data or fake.Since each cell receives a noise vector z, the formula for the probability of a phrase of length Tes given the noise vector z is: p (es — z) = p (w1 — z) T t = 2 p (wt — w lt; t, z) wt = arg max (Vht), where ht is the hidden state of cell t, where the t-th word determines wt. yt = We [wt] determines the output yt. In addition, this study uses approximate discretization to address the problem of the steepest descent method failing given a discrete problem. yt = We softmax (Vht-1 L) When L → , the term approaches yt = We [wt].

For training goals, the iterative optimization scheme consists of two steps. Minimize: LD = Es S log D (s) Ez pz (z) log [1 D (G (z))] Minimize: LG = tr (s1r + r-1s) + (s-µr) T (–s 1 + r-1) (µs-µr) where s and r represent the covariance matrix of the feature vectors fs and fr of the real and synthetic statements, respectively. µs and µr indicate the average vectors of fs and fr, respectively. s, r, µs, and µr are empirically estimated in mini-batch.

LSTM as discriminator:

The discriminator has the same structure as the generator, with the exception that it has a singlechannel output that represents the score. A sentence is represented by the matrix X RkT, where k is the length of the sentence and T is the dimension of the vector for one word. Wc Rkh is also applied to the sentence. In fact, the h has several sizes, and the concatenated results (denoted by feature vector f) are mapped to D(x) [0, 1] using a softmax layer. When the GAN concept was first proposed, the most widely utilized objective function was Eq. 28, in which x means real data, z means noise, D means discriminator network, and G represents generator. The Jensen Shannon (JS) divergence between the x and Gz distributions is improved by optimizing the objective function.JS divergence, on the other hand, has proven to be problematic as the distance cannot be displayed if the two distributions do not overlap.This can result in insecure training and mode collapse. (the generator is actually). Only some samples of data generation can be generated).[31] As a result,

Wasserstein GAN replaced JS divergence with Wasserstein distance (earth mover distance). This makes training more stable and eliminates the problem of mode collapse. The discriminator attempts to optimize the probability of identifying the actual data as "genuine" and the created data as "fake" according to a value function. The generator aims to reduce the possibility of flagging the data generated by the discriminator as "fake".

5.3.1 Forms

1. Firstly ,we built LSTM model for balanced data.For all possible level combinations, a balanced dataset has an equal amount of observations. The dataset used is IDMB reviews which is available in keras module. And the accuracy we got is around 98 percent which means classifier is well classified.

```
[ ] '''model=Sequential()
    model.add(Embedding(vocab_size,embed_size,input_shape=(x_train.shape[1],)))
    model.add(LSTM(units=60,activation='tanh'))
    model.add(Dense(units=1,activation='sigmoid'))
    model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
    model.summary()'''
    model = Sequential()
    model.add(Bidirectional(LSTM(20, return_sequences=True), input_shape=(x_train.shape[1],1)))
    model.add(TimeDistributed(Dense(1, activation='sigmoid')))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    model.summary()
    h=model.fit(x_train,y_train,batch_size=128,epochs=5)

    Model: "sequential_2"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     bidirectional_1 (Bidirectio  (None, 100, 40)          3520
     nal)

     time_distributed (TimeDistr  (None, 100, 1)           41
     ibuted)

    =================================================================
    Total params: 3,561
    Trainable params: 3,561
    Non-trainable params: 0
    _________________________________________________________________
```

Figure 5.1: LSTM classifier for balanced data

2. Next, we built Bi-directional LSTM classifier for the same balanced dataset (IDMB reviews).Unidirectional LSTM best preserves facts of the beyond due to the fact the best inputs it has visible are from the beyond. Using bidirectional will run your inputs in two directions: one from beyond to destiny and one from destiny to beyond. What sets this method apart from unidirectional is that within the LSTM that runs backwards, you can hold facts from the destiny, and by combining the two hidden states, you can hold facts from both beyond and destiny at any point in time. And the precision is 99.26 percent.

Figure 5.2: Bi-directional LSTM classifier for balanced data

3. As the accuracy on balanced dataset is 99 percent which is nearly 100 percent nothing can be done on that classifier. So we decided to change dataset i.e from balanced to unbalanced dataset. If the difference between positive and negative values is extremely large. We may then call our dataset Imbalance Dataset.

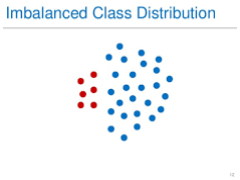


Figure 5.3: Imbalanced distribution

Figure 5.4: Balanced distribution

We took another dataset from this URL https://towardsdatascience.com/yet-another-twitter-sentiment-analysis-part-1-tackling-class-imbalance-4d7a7f717d44. It has around 90000 samples with three different classes namely positive, negative and neutral. We built LSTM on unbalanced dataset and got an accuracy of 94 percent. So after constructing confusion matrix we need to modify code to build generator.

Figure 5.5: LSTM classifier for unbalanced data

- As LSTM is giving higher accuracy even on unbalanced dataset , so we decided to make generator and discriminator as LSTM.
- We didn't use a Recurrent Neural Network since it has a problem with vanishing gradients, which implies it can't handle long-term dependencies.
- And the reason why we didn't choose Convolution Neural Networks is because they are good at image and speech classification.

Further Implementation

- It might be developed as an API to make it easier

5.3.2 Output Screens

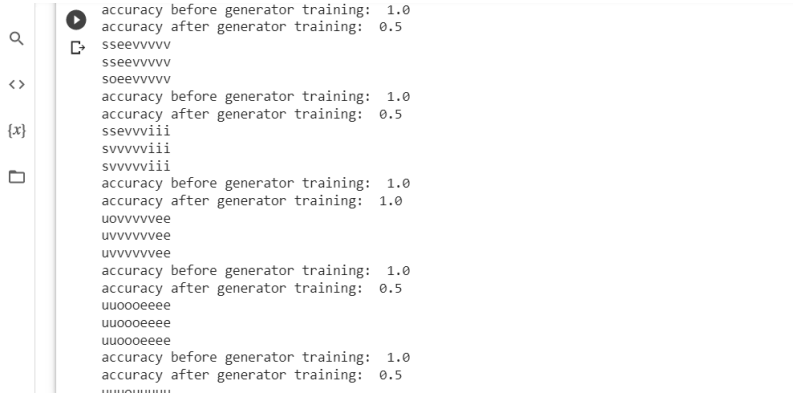


Figure 5.6: Output Screen 1

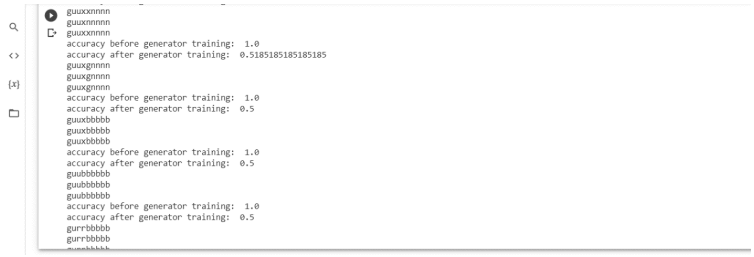


Figure 5.7: Output Screen 2

CHAPTER 6

Testing and Results

6.1 Testing

Generator:Create a fictitious pattern with the goal of providing false information to the discriminator.

Discriminator:By comparing each other, we try to distinguish between the actual and fake patterns

Repeat this process until you have a more accurate generator and discriminator.Mutual Validation is a typical method of evaluating text classifier performance. This works by dividing the training dataset into equal-length random sample sentences (for example, four sentences containing 25 percent of the data). [27]For each sentence, the remaining sample (for example, 75 percent of the sample) trains the text classifier. The classifier then makes a prediction for each sentence and compares the result to a human-annotated tag. This decides whether the forecast is correct (true positives and true negatives) or incorrect (false positives and false negatives) (false positives, false negatives). You can use these results to create performance metrics to help you quickly evaluate the performance of your classifier.

- **Accuracy**: Percentage of text classified with the correct tag.
  Accuracy=(TP+TN)/(TP+FP+TN+FN)

- **Precision**: Percentage of cases detected directly from the total number of cases predicted by the classifier on a particular day.
  Precision=TP/(TP+FP)

- **Recall**: Percentage of the total number of cases that the classifier should have predicted on a particular day.
  Recall=TP/(TP+FN)

- **F1 Score**: A balanced combination of precision and recall.

  F1 Score=2TP/(2TP+FP+FN)

Unstructured data makes up around 80 percent of all information, and language is one of the most common types of unstructured data. Because of the chaotic nature of language, analysing, understanding, organising, and sorting text data is difficult, time-consuming, and underutilised by most businesses. This is where text classification using machine learning comes in helpful. Text classifiers enable businesses to swiftly and economically arrange all types of relevant text, including email, legal documents, social media, chatbots, and research. Companies can save time evaluating textual data, automate business processes, and make data-driven business choices as a result of this technology.

6.2 Design of Use-cases

Text classification has dozens of applications and can be used for a variety of jobs. Data classification technologies are sometimes used behind the scenes to improve the functionality of apps that are used on a daily basis (such as email spam filtering). Marketers, product managers, engineers, and sales people utilise classifiers to automate business operations and save hundreds of hours of manual data processing in other circumstances.

Some of the main applications and use cases for text classification are:

- Identifying critical issues.
- Customer service processes are being automated.
- Listening to the Customer's Voice (VoC)

Customer service processes are being automated

One of the foundations of a healthy and thriving business is providing excellent customer service. 93 percent of organisations with exceptional customer service are likely to be repeat clients, according to Hubspot. According to the survey, 80 percent of respondents indicated they stopped doing business with the company due to bad customer service. Text classification assists support

teams in providing a positive experience by automating jobs that should be left to the computer and freeing up time for more critical work.[32]Text classification, for example, is often used to automate ticket routing and triage. Support tickets can be automatically routed to coworkers with specific product knowledge using text classification. When a consumer submits a refund request, the ticket can be allocated immediately to a teammate with the authorization to process the refund. This allows clients to receive a high-quality response more quickly. Emotion classification can also be used by support teams to automatically assess the urgency of support tickets and prioritise those that contain negative emotions. This helps to prevent customer churn and turn around negative situations.

Listening to the Customer's Voice (VoC)

Customers are listened to through surveys such as the net promoter score at every stage of the customer experience. The data is both qualitative and quantitative, and while NPS scores are simple to interpret, free-form replies require more in-depth analysis utilising text classification techniques. You can use machine learning to swiftly analyse candid customer comments instead of depending on people to analyse customer speech data. Classification models aid in the analysis of data to identify patterns and insights like:

- What do customers think of our products and services?
- What needs to be improved?
- What should I do differently?

Teams can make better judgments by combining quantitative and qualitative data instead of spending hours manually reviewing all open-ended responses.

6.2.1 Validation

Advantages of our implementation

- Deep learning (GAN) -based text classification methods can avoid tedious feature extraction processes and improve the prediction accuracy of large volumes of unstructured data benchmarking problems.

- Traditional text classification methods based on machine learning have many drawbacks, such as dimensional explosions and data.
- Lack, limited generalization capabilities, etc. continue. Therefore, use a deep learning model like GAN.

Disadvantages of our Implementation

- Without proper embedding and encoding levels in the LSTM, the model will not understand the actual meaning of the input string and will not provide the most accurate output class.

6.3 Result

We discovered that updating the discriminator twice as often as the generator enhances the model's performance. For each epoch, the generator gets major updates and reduces training time. NetTrain contains data generator functions that include "pattern" and "latent" vector generation. GAN is particularly unstable during exercise because it consists of two networks that interdepend on performance development. [33]Despite the many methods implemented to improve the stability of the GAN architecture, it still suffers from non-convergence. Early stopping strategies cannot obtain the best performing GAN model due to its unstable nature. EscaTo fix this, keep model

Table 6.1: Results

Classifier	Dataset	Accuracy
LSTM(Uni-directional)	Balanced	98
LSTM(Bi-directional)	Balanced	99
LSTM(Uni-directional)	Unbalanced	80
GAN	-	95

checkpoints and ngram performance metrics in each round and use the ngram performance metrics to select the best performing model. Little research has been done on the best hyperparameters for text GANs, so to evaluate the performance of our model, we will employ various hyperparameter combinations. As proposed by, the model is trained using the Adam Optimizer at a

given learning rate. Change the number of layers for both the generators and the critics, as well as the number of critic updates per generator update and the training set's maximum sentence length. I thought that by experimenting with several architectures, I would be able to identify the best mix for my objectives. [34]

CHAPTER 1
INTRODUCTION

CHAPTER 7

Conclusions and Future Scope

7.1 Conclusion

A novel adversarial training technique for text production was presented. We've shown that the suggested model outperforms the competition, can generate realistic words, and the learnt latent representation space can "smoothly" encode reliable words.The proposed method is compared with the basic model and the current method and evaluated quantitatively. Deep learning allows you to harness large amounts of computing power and data with little manual effort. GAN has greatly simplified the NLP problem by embedding words and using common deep learning models such as RNNs, CNNs, and VAEs. Researchers have many new chances to examine new text-generating applications thanks to significant increases in available data and processing power, as well as advances in deep learning. This tendency is projected to continue as more and better model designs become available. More NLP applications that use reinforcement learning methodologies to investigate text-writing studies are needed. Natural language generation is the goal of text generation research. Existing deep learning models do not yet fully comprehend the intricacies, strategies, and interpretations of natural language that build when generating larger texts. We need better indicators carefully constructed from human studies to gauge the success of a text generating model.

7.2 Future Scope

The code can be improved so that it can be extended and integrated with other systems. The code is currently implemented in Tensorflow. The main purpose is to train and test the model with structured data. It might be developed as an API to make it easier to use. Furthermore, while this tool

could be a big assistance in terms of creativity, we should consider how this model could be utilised for malicious ends.These are the kinds of concerns that need to be addressed right now. As the potential of AI technology grows rapidly, it is important to discuss how to best use AI technology.

References

[1] Gustavo H de Rosa and Joao P Papa. "A survey on text generation using generative adversarial networks". In: *Pattern Recognition* 119 (2021), p. 108098.

[2] Shayak Chakraborty, Jayanta Banik, Shubham Addhya, and Debraj Chatterjee. "Study of Dependency on number of LSTM units for Character based Text Generation models". In: *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*. IEEE. 2020, pp. 1–5.

[3] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. "Table-to-text generation by structure-aware seq2seq learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[4] Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. "Adversarial text generation via feature-mover's distance". In: *Advances in Neural Information Processing Systems* 31 (2018).

[5] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. "Long text generation via adversarial training with leaked information". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[6] Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. "Order-planning neural text generation from structured data". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[7] Jinyin Chen, Yangyang Wu, Chengyu Jia, Haibin Zheng, and Guohan Huang. "Customizable text generation via conditional text generative adversarial network". In: *Neurocomputing* 416 (2020), pp. 125–135.

[8] Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. "Neural text generation: Past, present and beyond". In: *arXiv preprint arXiv:1803.07133* (2018).

[9] Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. "Globally coherent text generation with neural checklist models". In: *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016, pp. 329–339.

[10] Ziang Xie. "Neural text generation: A practical guide". In: *arXiv preprint arXiv:1711.09534* (2017).

[11] Elizabeth Clark, Yangfeng Ji, and Noah A Smith. "Neural text generation in stories using entity representations as context". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 2250–2260.

[12] Liang Huang, Kai Zhao, and Mingbo Ma. "When to finish? optimal beam search for neural text generation (modulo beam size)". In: *arXiv preprint arXiv:1809.00069* (2018).

[13] Valerie Hajdik, Jan Buys, Michael W Goodman, and Emily M Bender. "Neural text generation from rich semantic representations". In: *arXiv preprint arXiv:1904.11564* (2019).

[14] Touseef Iqbal and Shaima Qureshi. "The survey: Text generation models in deep learning". In: *Journal of King Saud University-Computer and Information Sciences* (2020).

[15] Marco Damonte and Shay B Cohen. "Structural neural encoders for AMR-to-text generation". In: *arXiv preprint arXiv:1903.11410* (2019).

[16] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. "Learning neural templates for text generation". In: *arXiv preprint arXiv:1808.10122* (2018).

[17] Sandeep Subramanian, Sai Rajeswar Mudumba, Alessandro Sordoni, Adam Trischler, Aaron C Courville, and Chris Pal. "Towards text generation with adversarially learned neural outlines". In: *Advances in Neural Information Processing Systems* 31 (2018).

[18] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. "A generative model for category text generation". In: *Information Sciences* 450 (2018), pp. 301–315.

[19] William Fedus, Ian Goodfellow, and Andrew M Dai. "Maskgan: better text generation via filling in the_". In: *arXiv preprint arXiv:1801.07736* (2018).

[20] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. "The curious case of neural text degeneration". In: *arXiv preprint arXiv:1904.09751* (2019).

[21] Florian Schmidt, Stephan Mandt, and Thomas Hofmann. "Autoregressive text generation beyond feedback loops". In: *arXiv preprint arXiv:1908.11658* (2019).

[22] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. "Texygen: A benchmarking platform for text generation models". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 1097–1100.

[23] Shuang Chen, Jinpeng Wang, Xiaocheng Feng, Feng Jiang, Bing Qin, and Chin-Yew Lin. "Enhancing neural data-to-text generation models with external background knowledge". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3022–3032.

[24] Tianlu Wang, Xuezhi Wang, Yao Qin, Ben Packer, Kang Li, Jilin Chen, Alex Beutel, and Ed Chi. "Cat-gen: Improving robustness in nlp models via controlled adversarial text generation". In: *arXiv preprint arXiv:2010.02338* (2020).

[25] Weizhe Yuan, Graham Neubig, and Pengfei Liu. "Bartscore: Evaluating generated text as text generation". In: *Advances in Neural Information Processing Systems* 34 (2021).

[26] Yizhe Zhang, Zhe Gan, and Lawrence Carin. "Generating text via adversarial training". In: *NIPS workshop on Adversarial Training*. Vol. 21. academia. edu. 2016, pp. 21–32.

[27] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. "Adversarial feature matching for text generation". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 4006–4015.

[28] Weili Nie, Nina Narodytska, and Ankit Patel. "Relgan: Relational generative adversarial networks for text generation". In: *International conference on learning representations*. 2018.

[29] Haiyan Yin, Dingcheng Li, Xu Li, and Ping Li. "Meta-cotgan: A meta cooperative training paradigm for improving adversarial text generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 9466–9473.

[30] Pei Ke, Fei Huang, Minlie Huang, and Xiaoyan Zhu. "Araml: A stable adversarial training framework for text generation". In: *arXiv preprint arXiv:1908.07195* (2019).

[31] Heng Wang, Zengchang Qin, and Tao Wan. "Text generation based on generative adversarial nets with latent variables". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2018, pp. 92–103.

[32] Suphamongkol Akkaradamrongrat, Pornpimon Kachamas, and Sukree Sinthupinyo. "Text generation for imbalanced text classification". In: *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE. 2019, pp. 181–186.

[33] D Pawade, A Sakhapara, Mansi Jain, Neha Jain, and Krushi Gada. "Story scrambler-automatic text generation using word level RNN-LSTM". In: *International Journal of Information Technology and Computer Science (IJITCS)* 10.6 (2018), pp. 44–53.

[34] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. "A hybrid convolutional variational autoencoder for text generation". In: *arXiv preprint arXiv:1702.02390* (2017).