National College of Ireland

Neural Text-to-Text Generation System using Generative Adversarial Network and Monte Carlo Policy Gradient

MSc Research Project
Data Analytics

Seemanthini Narasimha Moorthy
Student ID: X18141447

School of Computing
National College of Ireland

Supervisor: Dr. Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Seemanthini Narasimha Moorthy
Student ID:	X18141447
Programme:	Data Analytics
Year:	2019-20
Module:	MSc Research Project
Supervisor:	Dr. Vladimir Milosavljevic
Submission Due Date:	12/12/2019
Project Title:	Neural Text-to-Text Generation System using Generative Adversarial Network and Monte Carlo Policy Gradient
Word Count:	5760
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	28th January 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Neural Text-to-Text Generation System using Generative Adversarial Network and Monte Carlo Policy Gradient

Seemanthini Narasimha Moorthy
X18141447

Abstract

Text-to-text generation is a fundamental task in natural language processing. Traditional models rely on standalone recurrant neural networks like Long Short Term Memory(LSTM) and Gated Recurrent Units(GRU). Generative Adversarial Networks (GAN) have found little success in generating discrete valued data like text. Major drawbacks lies in the failure to pass discrete output from generator model to discriminator model, and the inability of the discriminator model to assess incomplete sentences. This research strengthens the use of Generative Adversarial Networks combining it with Monte Carlo Policy Gradient, where the gradient policy update comes directly from the discriminator model and is passed back as the reward signal by using Monte Carlo Search algorithm. The results show that by combining Generative Adversarial Networks and Reinforce Algorithm, significant results can be obtained comparative to baseline models using evaluation metric called Bilingual Evaluation Understudy Score -N (BLEU-N). A BLEU score of 0.3 was achieved overall through different experiments.

Keywords: Natural Language Processing, Long Short Term Memory(LSTM), Gated Recurrent Units(GRU), Generative Adversarial Networks(GAN), Monte Carlo Policy Gradient

1 Introduction

1.1 Background and motivation

Language modelling has been a key aspect in natural language processing domain. It has further catapulted into advancement with the growth and development of deep learning. With innovative ways of text-to-text generation models emerging that matches state of the art, cross domain approaches are being applied to check if they can be tweaked to match the benchmarks of existing language models.Growth in big data has made quality text data corpus the need of the hour. Applications of natural language generation include text-to-speech generation,grammar and text correction, machine translation(Sutskever et al. (2011)), dialogue and poem generation and sequence prediction (Bahdanau et al. (2019)).

Neural text-to-text generation is the process of taking text data as input, and the model produces authentic text output. This is performed on the input text considering

several factors like structural integrity, syntactic and semantic analysis, and the overall meaning of the entire text corpus. Neural text-to-text generation language models have found positive success with traditional recurrent neural networks like long short term memory, variational autoencoders, gated recurrent units and Bidirectional Encoder Representations from Transformers (Devlin et al. (2018)). Goodfellow et al. (2014) introduced Generative Adversarial Networks (GAN) presenting a combination of two models: generator that creates new images from input and a discriminator that identifies if the text input is real (from input data), or fake (generated by the generator).

Reinforcement Learning is the area of machine learning in which an entity or agent is put in an environment and has to learn to accumulate reward points with suitable actions or get penalised for incorrect actions. Monte carlo policy gradient is a reinforcent policy gradient algorithm which used training samples to update the policy parameters. It considers the complete trajectory of training samples, reducing the gradient variance but keeping bias unaltered (Plaat et al. (2015)). Previous research shows the implementation of using GAN with reinforcement algorithms for neural text generation but had issues such as mode collapse, vanishing and exploding gradients. Generative adversarial network and reinforcement learning have succeeded independently in the past in varying degrees for text generation.

This research introduces a novel approach for neural text-to-text generation with the use of the successful image synthesis model Generative Adversarial Network. The drawbacks of GAN include difficulty in generation of discrete valued data and inability of discriminator to identify real versus generated text on the basis of partial output. These drawbacks will be worked on with the use of Monte carlo policy gradient with continuous reward feedback during generator training. The evaluation of newly generated/synthesized text output is evaluated on the basis of conventional language model metric like BLEU-N

The report is organised as the following sections: Section 2 provides the review on state of the art models and literature existing in the field of neural-text-to-text generation models, Section 3 discusses the methodology followed for the research implementation, Section 4 describes the design and implementation of the language modelling using GAN and Monte carlo policy gradient, Section 5 presents the evaluation of the implemented model and weighs in on the performance of the implementation with existing models, Section 6 presents the conclusion and future work.

1.2 Research Question

"Can text-to-text generation using deep learning be enhanced by combining Generative Adversarial Network and Monte carlo policy gradient in comparison to traditional Recurrent Neural Network models?"

2 Literature Review

This section consists of literature present in the domain of neural text generation. Numerous machine learning models have been implemented, which can mainly be put into three main categories.

2.1 Implementation using Recurrent Neural Networks

Recurrent neural networks have successfully been implemented for generation of discrete data like text and speech. Shedko (2018) utilised an RNN long short term memory model for text generation for synthesizing Baron style poem creation using Gutenberg text corpus. The text synthesis succeeded, with the emotional and author style mimicry carried out with the help of traditional bag of words model. This work could be improved upon with the addition of after-processing network model to improve output. Zhang et al. (2019) proposed the Self Labelling Conditional Variational AutoEncoder consisting of a labelling model network indicating the perfect encoder for the selected decoder. This experiment was conducted on EGOODS dataset. There were several models tried with the dataset, but it worked best for one to many text generation. The quality of text improved with the inclusion of encoder-decoder model, offering similar accuracy scores in comparison to baseline sequence-to-sequence models. Similar work has been carried out by Xu et al. (2018) called TextDream where semantic space search is performed with initial randomised seed input. This performs better in comparison to conditional variable auto-encoder model as semantic search performs better than text generated using single tokenized label. The ability of text generation switching to different fields indicates the diversity of TextDream model and is reflected in the discinct text output scores. Ruan et al. (2019) went a step further and proposed Condition-Transforming Variational AutoEncoder- a VAE model with a latent variable taken as input and non-linear transformation performed on it by encoder. This model has good evaluation scores for the subjectivity by providing topic-relevant responses and domain ease.

There have been research performed where the implementation works on multiple tasks for the same dataset. Wang and Wu (2018) utilised VAE for binary classification and text generation tasks. Long short term memory with self-attention for the enhancement of autoencoder output. The objective of the experiment was successful with fixed length and compact text generated resonating the meaning of the input corpus. The only reason this could not compete with the baseline DCNN and SVM model are the vanishing KL problem and high error rate. Chen, Wu, Jia, Zheng and Huang (2019) is another sequence to sequence model in which headlines generation is considered as a summarization task with Keyword Enhanced Diverse Beam Search(KEDBS), where a CNN-RNN model is used to generate semantically relevant text output with altogether good ROUGE scoring. Wang et al. (2019) implemented a conversational dialogue agent where dialogue history is integrated with sequence-to-sequence convolutional neural network model providing external knowledge to gain relevant text output. The Ubuntu dialogue corpus was used for this model. Absence of provided background information in the generated output and constant variability with fluctuation resulted in lower BLEU scores as less as 0.07. Budhkar et al. (2019) utilised a word embeddings model presented at train time. The advantage of this model is the irrelevance of vocabulary size in generating text output. With random noise fed into the discriminator, classification is done on the basis of it being real or artificial. Discrete nature of text input is ignored by the use of semantic

information.

Recurrent neural networks have a few problems despite good results like gradient explosion and vanishing which have been mitigated with GAN methods for text generation.

2.2 Language modelling using Generative Adversarial Network

Goodfellow et al. (2014) introduced the GAN model with impeccable abilities for image synthesis. Several researches have been conducted on implementation of GAN in the field of natural language processing. Li et al. (2018) introduced Text-to-Text GAN which recreates paraphrased sentences of word embeddings from continuous space input. Words with closest meaning to the input text space are replaced in the paraphrased output, retaining its core semantic meaning. This appears to create authentic sentences. Significant text corpus used in this implementation include Chinese and English movie reviews. High output text diversity was displayed in Chinese, with good ROUGE scores and the adaptability nature of model to multiple languages. Remarkable changes in this research includes word embedding comparison with output text using euclidian distance, which seamlessly produces continuous text output from input. Fedus et al. (2018) proposed a popular language model named Conditional-GAN, with the principle functionality of blank with a word of similar meaning, gauging some knowledge of words surrounding the blank space. This process is termed as masking. The hidden state is filled with the meanings of words the model has come across until the time step. Typical datasets like Penn treebank and IMDB movie review datasets were used to train and test this implementation. This implementation performs better on hidden vector space encoding in comparison Yu et al. (2016) with a significant increase of 6% in ROUGE scores.

Chen, Li, Jin, Zhang, Dai, Chen and Song (2019) introduces a model which is similar to Yu et al. (2016) SEQGAN model, sans the off-policy policy gradient. The generator model sequentially generates part of the output with the ability to multitask, i.e. work on the complete and partially generated output at the same time. Sub-samples are taken before the complete output is evaluated, in comparison of halting the reward evaluation system until the complete output is obtained.One drawback of sub-sequence sampling is the sub-sequence vector being sparse. Due to the sub-sequence sampling, the discriminator receives both complete and partial output samples provides authenticity prediction. This implementation has an edge over other models due to better feedback count and better feedback analysis of partial outputs. This reduces the risk of mode collapse with smaller text outputs in comparison to the scenario where the discriminator needs to wait for the entire output to be generated, and proportional evaluation time being indicative of discriminator output. This also improves long output dependency. EMNLP2017 WMT news dataset is used for this research.

Li et al. (2015) describe the model as a conversational system with the use of an objective function named Maximum Mutual Information (MMI). MMI extrapolates the information between source and destination, magnifying the knowledge held on during training. Twitter conversation and Opensubtitles are the datasets used for training, keeping in mind the conversational nature of the model. The model is rewarded poorly for uninteresting responses. Evaluation metrics used are BLEU scores and human assessment. Zhang et al. (2017) introduced TextGAN, where long short term memory RNN model is used as generator and convolutional neural network model is used as discriminator. A high dimentional feature list is presented with a discrepancy metric to which

the kernel is applied, which matches authentic and synthetic data. The moments in generated output is matched to the latent vector space, making the moment matching a kernel-based approach. Improving long output behavior is the one drawback of this model. Lin et al. (2017) introduces RankGAN in which appropriate ranks or positions are presented to both machine generated and human written texts in place of bifurcating the data set and limiting it to a binary grouping classification. Chinese poem dataset and Shakespeare's text corpus are fed into the model. As the input text contains more than two ranks, the discriminator has a better probability of recognising synthesized output by giving it a broader berth at recognition., this in return helps generator create better and authentic text outputs.

Until this point, there are several models which produces syntactically and semantically understandable outputs. There is a lack of the emotion and feeling that comes with a real human's work which lacks in artificially generated text. Wang and Wan (2018) introduced SentiGAN, with a complex architecture containing multiple generators and one multi-class discriminator model. Several generators are provided with unsupervised training to produce sentimental text. Every generator is allocated a sentiment, with multiple datasets being worked on at the same time. In comparison to variational auto-encoder and SeqGan, SentiGAN performs better with almost 5% higher accuracy. This model provides compact sentences filled with the required sentiments, coming very close to mimicking human emotion. Guo et al. (2017) has an implementation similar to Fedus et al. (2018) as the discriminator purposefully seeps out information about its knowledge on current time step learnings to the generator model. This strengthens the generator output in the future. The concept of manager and worker modules are introduced, where the additional seeped information from the discriminator to generator is passed onto the manager module and the worker module receives the features that have been extracted from the leaked information. A big advantage of this implementation is that unsupervised training can be performed on the manager and worker module to produce text. Haidar and Rezagholizadeh (2019) implemented TextKD-GAN brought out a combination model containing variational auto-encoder and generative adversarial networks. This is a knowledge distillation approach in which weakened information is moved from the VAE model known as teacher model to the generator known as the student model. The one hot encoded representation from real data is compared with the softmax output of the student model, which nullifies the necessity of pretraining. GAN generators function best with pretrained settings, but due to its absence, the VAE is pretrained. Vanishing gradient is the main drawback of this implementation.

Despite the success of generative adversarial networks in neural text generation, there are still multiple shortcomings with regards to long text output generation, holding syntactic and semantic integrity when compared to input text, where reinforcement algorithms fare better when combined with GAN.

2.3 Implementation using Generative Adversarial network with Reinforcement learning algorithms

Reinforcement learning policy gradient algorithms alleviate several issues existing in pure GAN text generation models. Appropriate actions performed by the agent is rewarded, else penalised. Reward optimisation is performed best by policy gradient algorithms in comparison to other implementations of reinforcement learning. There are several works illustrating advantages of combining the novel concepts. Li et al. (2017) introduced a

dialogue generation system for public domain utilising reinforcement learning. The discriminator classification output is fed back to the generator as usual. REGS or Reward for Every Generation Step is calculated at every step, and adversarial reward estimation and evaluation is performed. Credit assignment is done for mean-squared loss for every pair of authentic and estimated reward. Monte carlo tree search is used for intermediate reward assignment and rewards for discriminator are restricted sequences used for decoding with the help of teacher training. This model has 18% better accuracy when compared to plain reinforcement language model. The model which presented itself as a baseline model for future language models is the Sequential Generative Adversarial Network or SeqGAN. This model makes use of a typical GAN model with reinforcement learning to improve throughput. BLUE score is the metric used to evaluate quality of output text generated. This model comparatively has a higher BLEU score with reference to other baseline models. Political speeches of previous American president Mr. Barack Obama has been used along with quantrains in Chinese language for testing compatibility and variability with change in language. The model failed to provide lengthy outputs, which happens to be a major drawback. Xu et al. (2019) implemented a model where feature learning for the successor happens during training with the help of reinforcement learning. The model also includes token generator which is based on maximum likelihood estimator. The incomplete output is taken and the estimator determines the next value to be filled in for the output to be complete. The value function input is divided into two parts: a successor map and reward predictor which works on the current and future time step word prediction at the same time. This is the major difference between this implementation and other implementations with reinforcement learning. Vanishing and exploding gradients were still major issues that were left unresolved with this proposal.

Rajeswar et al. (2017) used the GAN objective function as the baseline, and the proposed the discriminator network provide continuous valued input producing discrete outputs with syntactic and semantic accuracy executed on the Chinese poem dataset. The LSTM generator model BLEU score of 87% is the highest in comparison to other baseline models. Zheng et al. (2017) introduces a comment generation model for news websites, preserving the semantic relevance of both news and previous comments. With knowledge of sentiments present, this shows a similarity to the implementation by Wang and Wu (2018)'s SentiGAN. Gated attention mechanism is implemented, with randomised samples being fed the generator for diversity in generated comments. Adversarial learning is introduced to this model with the inclusion of reinforcement learning. The news comments are web scrapped from the news website 163.com, which is used as the input text corpus. Word and character level text generation models are implemented. Performance degradation is the major pitfall of this model, Where diversity in sentiments makes the discriminator fail to classify comments. The solution to this could be to combine both word and character level modelling. citeLi20182 implements a model named CS-GAN in which RNN generator is assembled with reinforcement learning algorithm. With the knowledge of current character input and input semantic space, the generator has to predict the next suitable character for the output. The main advantage of this model is the use of categorical information stored to make labelled data. RNN helps in learning and retaining the sentence structure. Low volume data obstructs the performance of the supervised learning model.

3 Methodology

Several methodological approaches exist for machine learning problems. Looking into the natural language processing and language modelling aspect of the implementation, Cross-Industry Standard Process for Data Mining also known as CRISP-DM is gauged to be the optimal approach that provides the full understanding of this research and its objectives. All phases and steps for the methodology are visualised in Figure 1

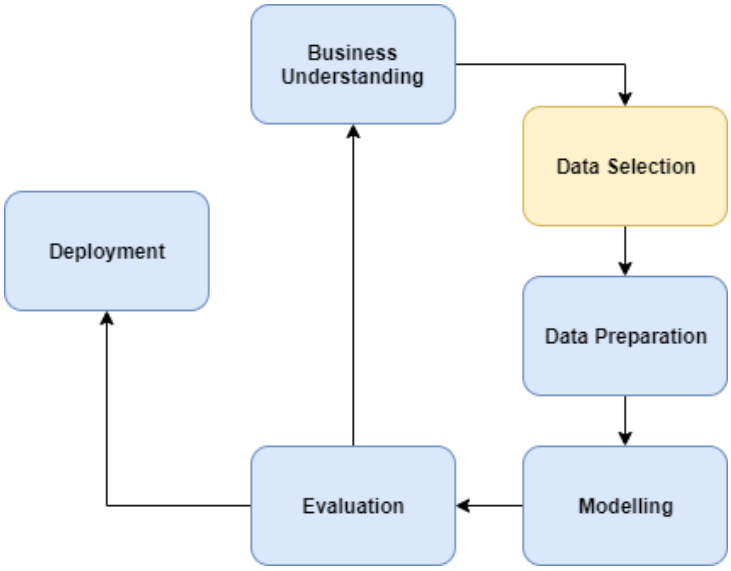

Figure 1: CRISP-DM methodology

3.1 Business Understanding

Neural text generation has been existent in some form or the other since the internet was created. Graves (2013) has implemented traditional RNN for long text generation with text input from Shakespeare's work. Not just in the literary world, text-to-text generation systems are also faring well in the tech industry with deep learning models generating latex and C code from sample input codes. It has been stretched to as far as generating baby names, with considerable success. [1]. The process of software learning to imitate input style reflects upon the human learning process. Humans read novels, news and other literary articles. They understand how sentences have to be constructed and words morphologically placed to present to the world the best group of words together that shows how they are feeling.

Text-to-text generation has been used in news reporting as well where Dušek et al. (2020) talks a "robot journalist" that reported a small earthquake within three minutes of its occurrence with details like intensity, location and time. Criticality of the situation makes to the data being generated faster at a higher stakes. Rival newspapers took more time than a software whose work was to publish information on events as and when it is happening.

Language models could save human workload and create textual or literary work in a considerably less amount of time. This elevates the humans to utilise their time

[1] http://karpathy.github.io/2015/05/21/rnn-effectiveness/

supervising the software and give inputs for creativity and emotions, where the software clearly lags behind.

3.2 Data Selection

Due to the nature and complexity of language modelling where the model implementation has to have the ability to understand the features of input text, data selection plays a bigger part in comparison to data understanding. Hence, data selection plays a key role in this implementation. In text-to-text generation problems, it is important to have large data corpus for the model to thoroughly understand the nature, syntactic and semantic complexity of the output text to be generated. Gutenberg website [2] is an online open source digital library with over 60,000 books in various languages. This implementation will be working on literary works written in English. Lahiri (2014) has provided a portion of the Gutenberg dataset after performing some pre-processing steps such as removing data from the digital articles not relevant for model input like meta data. This dataset has a total of 3037 novels from 142 authors. Looking at the selected data from an ethics perspective, the Gutenberg dataset has been used in multiple language model implementations, as it happens to be a open source dataset. Digital works of literature including fiction and non-fiction with over 60,000 books is publicly available. Due to the absence of personal information, it is GDPR compliant.

3.3 Data Preparation

The data corpus obtained from Lahiri (2014) has taken care of the pre-processing tasks including removal of metadata, author license information and transcription notes in each literary work. Due to the nature of language modelling, the punctuation are retained to maintain the quality of output text generation. The pre-processing and data preparation steps are explained below:

- After anaylsis, unidentified or wrongly encoded characters were found to be absent
- All characters were converted to lower case for uniformity
- Tokenization: It is the process of taking the text corpus and breaking it down to distinct individual words. The token '_BEGIN' is added to mark the beginning of tokenization vector.
- Word embedding: Tokenised words are mapped onto real number vector representation, also known as indexing , where each word has a corresponding real number which acts as it's index. This is useful during output generation for mapping the index back to word

3.4 Modelling

Unsupervised deep learning is the data mining algorithm implemented in this research due to the selection of unstructured/ unlabelled data for the purpose of neural text generation. The research implementation has 2 major components:

1. Generative Adversarial Networks

[2] https://www.gutenberg.org/

2. Monte Carlo Policy Gradient

3.4.1 Generative Adversarial Networks

The generative adversarial network has two main components:

1. Generator: A deep learning model which takes input, studies its features and attributes and provides output which tries to mimic the input. This works well with unstructured data like images, text and speech.

2. Discriminator: A deep learning model which receives input and returns either true or false based on authenticity. It is fed with data from the generator (synthesized input) and data from text corpus(real input) in random order. Due to this, discriminator helps in creating better outputs from generator.

In this implementation, a variation of recurrent neural network called Gated Recurrent Unit (GRU) has been used as both generator and discriminator.

Gated Recurrent Unit : Proposed by Hendrickx et al. (2015), GRU is a type of RNN with two gates: an update gate and reset gate. These gates work as the vectors holding information that has to be passed onto the output. Data retention is high and eliminates the necessity to manually remove irrelevant data with multiple time steps. The update gate decided the proportion of present input to be passed onto the future. The reset gate determines the proportion of data to be eliminated with passing time steps. GRU does not require memory units, making it easier to train.

3.4.2 Monte Carlo Policy Gradient

Policy gradient algorithms are often used in reinforcement learning implementations. The optimisation of expected rewards drives the agent to avoid getting penalised. To achieve this, the overall rewards should be equal to the product of a trajectory step and the corresponding step reward. The state and actions depends on the task at hand. The objective for this implementation is to provide the best text output by the generator.

Monte carlo rollouts is the algorithm which plays out the entire scenario considering the past and current states, to see the total reward obtained in the end. This calculates the exact value of the trajectory predicted for the current time step. This shows monte carlo rollouts have high variance and zero bias, because even a small change in the input can change the entire course of the future for reward estimation (Browne et al. (2012)).Plain GAN model for neural text generation allows the discriminator to provide unsupervised training to the generator. To reduce the variance during reward assignment, generator training alternates between supervised and unsupervised.

3.5 Evaluation and Deployment

After data modelling, it is necessary to evaluate the results presented by the research. Evaluation of results with current text input gives better understanding for business understanding over future iterations. The metric being used for evaluation are as follows:

3.5.1 BLEU Score

BiLingual Evaluation Understudy or BLEU is the evaluation metric for various natural language processing problems. First introduced by Papineni et al. (2002), it rose to popularity with machine translation and summarisation tasks. BLEU measures the number of n-grams in the output(neural text generation model output) that also appears in the input text. BLEU is similar to precision metric, as a reference in the output is tried to be found in the input. In simple words, it gives a measure of how close a model text output is to the input text. BLEU score can range from 0 to 100, 100 being the score of sentence very close to meaning with input sentence.

4 Design and Implementation

The implementation is the most most important aspect of any research. The literature review has indicated some imporvements over existing models, which will be experimented in this implementation.
The overview of design architecure for the implementation is depicted in Figure 2 below:

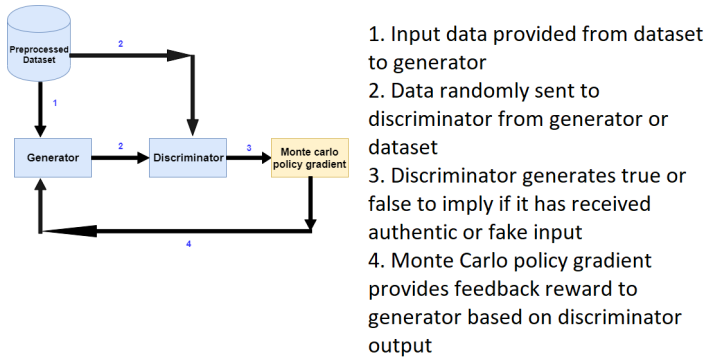

Figure 2: Architecture design

The implementation is performed using python on Anaconda Spyder because the IDE conveniently helps in training the model by providing all necessary packages. Tensorflow is the open source library used for the deep learning tasks and programming. [3] The dataset provided by Lahiri (2014) is utilised in varying forms for model training. The input files are created based on case studies and fed as input to the model. All the necessary installations and packages are already done before model training. As mentioned in Section 3.3, the input file is converted to lower case, tokenised to create an integer index vector for ease of output generation. Tensorflow works best with numbers rather than text. Tokenisation helps in optimising the model performance. A '_BEGIN' token is added to mark the beginning of the input file. Before feeding the processed data to the model, a preliminary sanity check is performed on the tokenised dataset to check the number of tokenised characters, and number of bigrams in the dataset. A bigram is the number of unique pair of words which will be used for evaluation of output using BLEU score.BLEU scores have various forms like unigram(BLEU-1), bigram (BLEU-2), trigram(BLEU-3) and N-Gram(BLEU-N) scoring. For this implementation, BLEU-2 score will be used for evaluation.It checks for the number of bigrams in the output that

[3] https://www.tensorflow.org/

are present in the input. The data is then fed into the GAN-monte carlo policy gradient model for training and the overview of process flow is depicted in Figure 3:

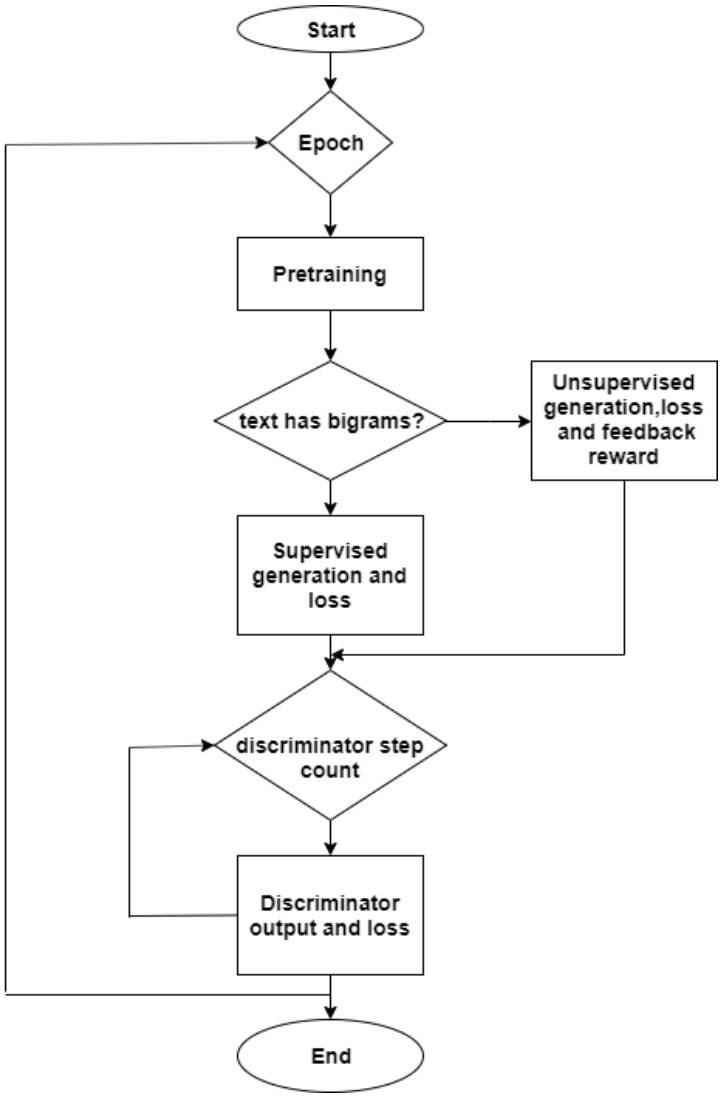

Figure 3: Execution flow

The generator and discriminator are alternatively made to optimise their objectives. Monte carlo policy gradient helps the generator in selecting the most appropriate discrete valued character at each time step.Through Figure 3, we see that with every epoch, the pretraining is performed which lets the generator produce output. Words are formed character by character, and for each epoch the character limit is set to 100. Due to the textual nature of input, embedding dimension has to be mentioned for the deep learning model. Embedding dimension is used after tokenization, being used during generation process. After pretraining generation, presence of bigrams leads the model to perform supervised learning of generator. Absence of bigrams leads to the Monte carlo policy gradient to be activated which provides the feedback reward and unsupervised learning is performed on the generator. Bigram detection is the standard set for sentence quality, and is proportional to the number of times Monte carlo policy gradient will be applied on the generator. After supervised or unsupervised generator step, discriminator training is performed. The discriminator step count is kept configurable to check model performance under varying discriminator steps. Discriminator model will be trained depending on the

number of discriminator steps. With every step, the discriminator gives the output of true or false. True output carries the meaning that the discriminator thinks the input provided comes from real dataset, and false means the input was synthesized by the generator. Discriminator loss depends on the correctness of the logits set by its output.
The epocs determine number of training steps for generator and discriminator. Another parameter used is the switch rate, which helps in tweaking the switch between supervised and unsupervised training. Adam optimiser is used for both generator and discriminator optimisation.[4] After the training, the discriminator, superviser generation and unsupervised generation training loss is plotted per epoch. This helps in model training progress and validation. As this is unsupervised learning based model, there is no separate validation or test step. The trained model is checked for output generation and output text quality of evaluated using BLEU scores. BLEU scores are calculated for every output text generated per epoch. The BLEU scores for supervised text generation and unsupervised text generation is plotted per epoch.
The configurable parameters to check model performance are epochs and discriminator steps, which have been discussed in the next section.

5 Evaluation

The model parameters are tweaked as per the case studies, which will be discussed in detail in this section. The dataset is large and is divided in two ways for performance evaluation:

- Full file: All 3037 novels are consolidated into a single file
- Half file: 1513 novels are consolidated into a single file

5.1 Performance based on discriminator step count

This experimental case study is performed to check the if the discriminator step count variations has any affect on model performance.

The full file has been considered for this study. For checking the magnitude of variation, the epoch has been considered in proportion to discriminator step count. First setup takes the full file with 140 epochs and 3 discriminator steps. The second setup takes the full file with 300 epochs with 6 discriminator steps. The smaller epoch is paired with smaller discriminator step. From Figure 4 in the training loss graphs, we observe that the supervised generator loss has stabilised to a value around 2, and the unsupervised generator and discriminator loss has consistently remained low despite minor fluctuations. The BLEU score graphs show the highest BLEU score obtained for both setups is around 0.3.

[4]https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

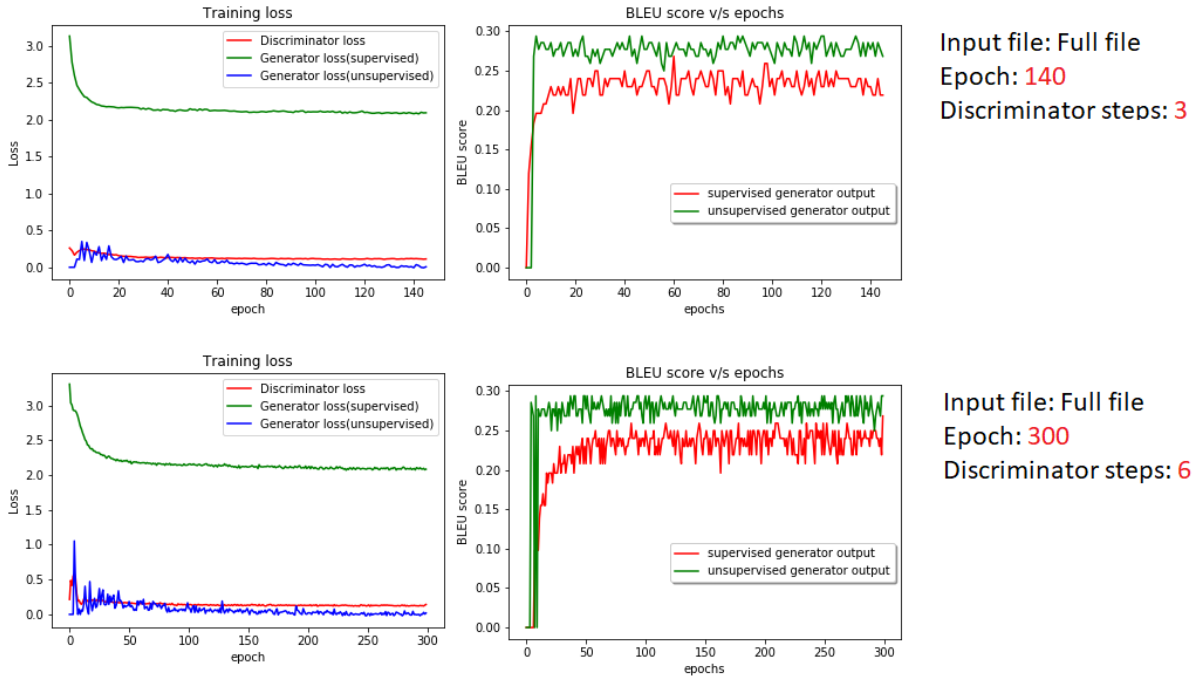

Figure 4: Model performance with varying discriminator step count and epochs

5.2 Performance based on input text size with respect to epochs

For this study, a slight variation has been introduced. The size of the file is considered inversely proportional to the epoch size. After executing the model training, it was noticed the supervised generator failed at epoch 50 consistently as seen from Figure 5. Several tweaks were performed for other parameters like switch rate and discriminator steps. The root cause of this generation failure was mode collapse at epoch 50. The mode collapse could be overcome by reducing the switch rate. Initially, the two models for this use case were run on switch rate of 0.2, but after checking with smaller values, the optimal switch rate with considerable output text happened with switch rate of 0.001 as seen in Figure 6. With the switch rate set to 0.2, we see the the supervisor generation loss drop, whereas unsupervised generator and discriminator low stayed low as seen in above study. With the full file, we see that BLEU score almost constantly has BLEU score of 0.3 for unsupervised generation and goes up to 0.2 for supervised generation. For the half file, once again the unsupervised output gives BLEU score of 0.3 but the supervised output has a stabilised score of 0.07 till epoch 20, then raises till 0.15 and falls to zero. When the switch rate is changed to 0.001, we see there is not much change in the loss graph other than the fact that the mode collapse has been eliminated. The BLUE score shows some consistency with both supervised and unsupervised output. Table 1 shows samples of output generated, and maximum BLEU score is obtained with this experiment.

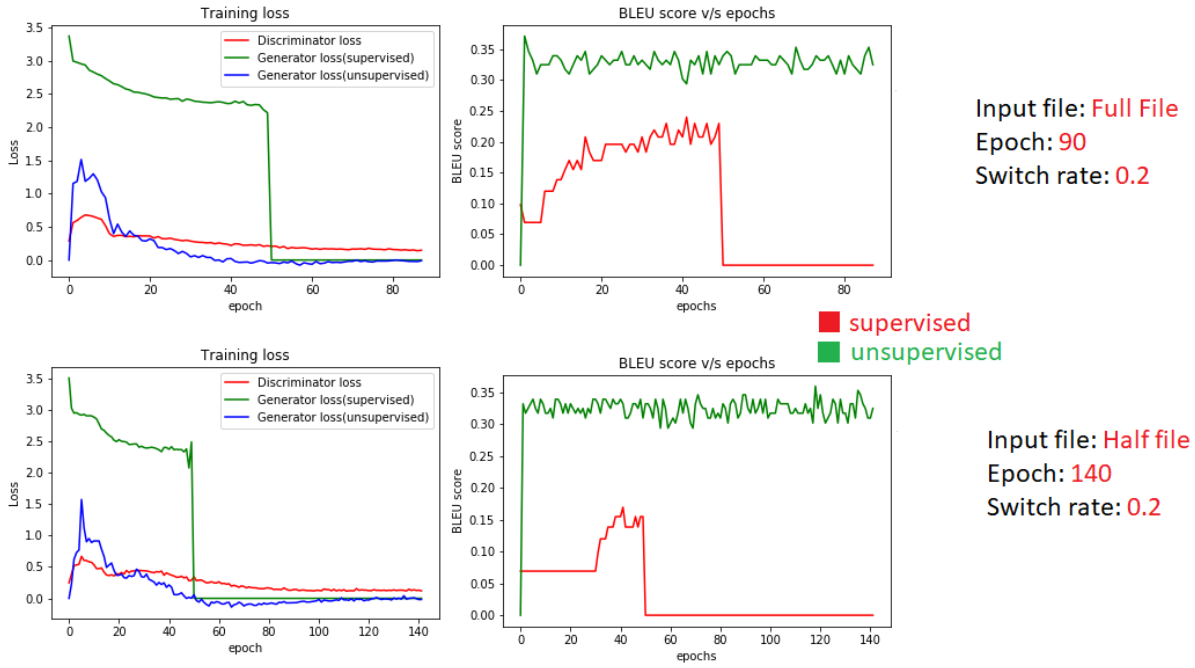

Figure 5: Model performance based on file size, epoch and switch rate

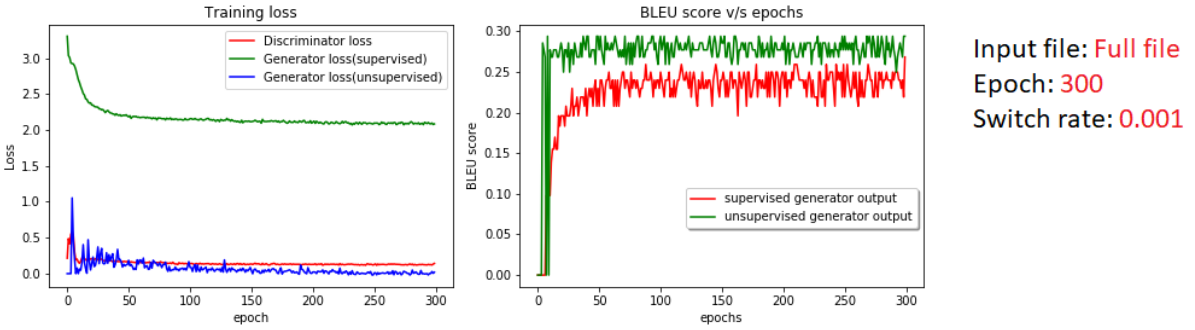

Figure 6: Model performance based on file size, epoch and tweaked switch rate

6 Discussion

Overall evaluation shows BLEU score of 0.3 is obtained through various experiments. Table 1 gives a few sample outputs obtained through the experiments along with their BLEU scores. In the first setup, with the variation in discriminator steps,Figure 4 shows there is not much change observed in model performance, other than minor fluctuations. With the second setup as seen from Figure 5, we see there is mode collapse irrespective of input file size. This is seen to be rectified by changing the switch rate of supervised and unsupervised training of generator. The changes made through mode collapse is seen in the improvement in BLEU score as observed in Figure 6. We could also observe that the unsupervised output text consistently have BLEU scores more than supervised output text. This is directly the effect of using MCPG, with constant rewards being sent to the unsupervised generator step. We also notice that the generator discriminator switch does not affect the loss of either generator or discriminator. With the sanity search of bigrams in input text, we get clarity of what to expect in the output text. We observe from Figure

Table 1: Text outputs generated through experiment

Case	Text output	BLEU score
1	the doopare kiny, nonse asuine fabloniop of yound, whander, amaled to the, to hele wond herserys, e	0.297
2	uuirs, glir tore had she bangsed teure they of this, sound hirdm stoopk haing ady bis sousanteed, wa	0.294
3	eeer ind that lalcey tueco as doncancon tho t"e boichon the wechz merset morses ans to ad the seseve	0.23
4	5rsirpahuy w ewn ioe a,mc ai snfhtoec itagnssre fcm ree wia wean a"ooh inehdfasrsmtgeedncslehyt l	0.24
5	bec was the boays hi lumt rraney to theis hint, hathed fure rinting was tabayter a tereded a seailm	0.317

6 that we obtain the best results with big data input and epoch and small switch rate.

7 Conclusion

This research has implemented a text-to-text generation model using an image synthesis method known as Generative adversarial network with Monte carlo policy gradient. The implementation was performed on a small subset of Gutenberg dataset. The model has succeeded in overcoming the major drawback of the reason GAN fail to work with discrete data. The implementation of reinforcement learning algorithm has proven beneficial for much higher BLEU scores in comparison with regular supervised text output scores. The Monte carlo policy gradient has shown considerable improvement in text output generation with unsupervised genreation BLEU score always being higher than supervised output BLEU scores.With 2 case studies showing performance of model with discriminator step count and effect of input file size, epoch count and switch rate, we see better outputs when reinforcement algorithm is applied. The BLEU score of 0.3 was obtained overall, proving this is a moderately good text generation model.

7.1 Future Work

Altogether, the implementation has shown moderate performance considering the data corpus and other factors. Future work could include working with larger and more diverse datasets. It can also be extended to non-literary text generation like programming code generation. Currently, this model has been implemented based on Monte carlo reinforcement algorithm, future works could consider use of other on-policy and off policy reinforcement algorithms. More control over switch rate between supervised and unsupervised generation could be brought in to avoid mode collapse.

Acknowledgements

I would first like to thank my thesis advisor Dr. Vladimir Milosavljevic. He has constantly allowed this research to be my own work, led me in the right direction and gave valuable feedback and advice. I would also like to thank my family, partner and friends for their undying support.

References

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Courville, A., Pineau, R. L. J. and Bengio, Y. (2019). An actor-critic algorithm for sequence prediction, *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2015): 1–17.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. and Colton, S. (2012). A survey of Monte Carlo tree search methods, *IEEE Transactions on Computational Intelligence and AI in Games* **4**(1): 1–43.

Budhkar, A., Vishnubhotla, K., Hossain, S. and Rudzicz, F. (2019). Generative Adversarial Networks for text using word2vec intermediaries, (2).

Chen, J., Wu, Y., Jia, C., Zheng, H. and Huang, G. (2019). Customizable Text Generation via Conditional Text Generative Adversarial Network, *Neurocomputing* (2019).

Chen, X., Li, Y., Jin, P., Zhang, J., Dai, X., Chen, J. and Song, G. (2019). Adversarial Sub-sequence for Text Generation, pp. 1–10.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, (Mlm).
URL: *http://arxiv.org/abs/1810.04805*

Dušek, O., Novikova, J. and Rieser, V. (2020). Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge, *Computer Speech Language* **59**: 123 – 156.
URL: *http://www.sciencedirect.com/science/article/pii/S0885230819300919*

Fedus, W., Goodfellow, I. and Dai, A. M. (2018). MaskGAN: Better Text Generation via Filling in the_____.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative Adversarial Networks, pp. 1–9.

Graves, A. (2013). Generating sequences with recurrent neural networks, *CoRR* **abs/1308.0850**.
URL: *http://arxiv.org/abs/1308.0850*

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y. and Wang, J. (2017). Long Text Generation via Adversarial Training with Leaked Information.

Haidar, M. A. and Rezagholizadeh, M. (2019). TextKD-GAN: Text Generation using KnowledgeDistillation and Generative Adversarial Networks.

Hendrickx, T., Cule, B., Meysman, P., Naulaerts, S., Laukens, K. and Goethals, B. (2015). Mining association rules in graphs based on frequent cohesive itemsets, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **9078**(3): 637–648.

Lahiri, S. (2014). Complexity of Word Collocation Networks: A Preliminary Structural Analysis, *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, Gothenburg, Sweden, pp. 96–105.
URL: *http://www.aclweb.org/anthology/E14-3011*

Li, C., Su, Y. and Liu, W. (2018). Text-To-Text Generative Adversarial Networks, *Proceedings of the International Joint Conference on Neural Networks* **2018-July**: 1–7.

Li, J., Galley, M., Brockett, C., Gao, J. and Dolan, B. (2015). A Diversity-Promoting Objective Function for Neural Conversation Models.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A. and Jurafsky, D. (2017). Adversarial Learning for Neural Dialogue Generation.

Lin, K., Li, D., He, X., Zhang, Z. and Sun, M.-T. (2017). Adversarial Ranking for Language Generation, (Nips).

Papineni, K., Roukos, S., Ward, T. and Zhu, W. J. (2002). Bleu: a method for automatic evaluation of machine translation.

Plaat, A., Van Den Herik, J. and Kosters, W. (2015). Advances in computer games: 14th international conference, acg 2015 leiden, the netherlands, july 1-3, 2015 revised selected papers, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **9525**: 1–11.

Rajeswar, S., Subramanian, S., Dutil, F., Pal, C. and Courville, A. (2017). Adversarial Generation of Natural Language.

Ruan, Y.-P., Ling, Z.-H., Liu, Q., Chen, Z. and Indurkhya, N. (2019). Condition-transforming Variational Autoencoder for Conversation Response Generation, pp. 7215–7219.

Shedko, A. Y. (2018). Semantic-map-based assistant for creative text generation, *Procedia Computer Science* **123**: 446–450.

Sutskever, I., Martens, J. and Hinton, G. (2011). Generating text with recurrent neural networks, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011* pp. 1017–1024.

Wang, K. and Wan, X. (2018). Sentigan: Generating sentimental texts via mixture adversarial networks, *IJCAI International Joint Conference on Artificial Intelligence* **2018-July**: 4446–4452.

Wang, Z., Wang, Z., Long, Y., Wang, J., Xu, Z. and Wang, B. (2019). Enhancing generative conversational service agents with dialog history and external knowledge, *Computer Speech and Language* **54**: 71–85.

Wang, Z. and Wu, Q. (2018). An Integrated Deep Generative Model for Text Classification and Generation, *Mathematical Problems in Engineering* **2018**: 1–8.

Xu, C., Li, Q., Zhang, D., Xie, Y. and Li, X. (2019). Deep successor feature learning for text generation, *Neurocomputing* (xxxx).

Xu, W., Sun, H., Deng, C. and Tan, Y. (2018). TextDream: Conditional Text Generation by Searching in the Semantic Space, *2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings* pp. 1–6.

Yu, L., Zhang, W., Wang, J. and Yu, Y. (2016). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.

Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D. and Carin, L. (2017). Adversarial Feature Matching for Text Generation.

Zhang, Y., Wang, Y., Zhang, L., Zhang, Z. and Gai, K. (2019). Improve Diverse Text Generation by Self Labeling Conditional Variational Auto Encoder, pp. 2767–2771.

Zheng, H. T., Wang, W., Chen, W. and Sangaiah, A. K. (2017). Automatic Generation of News Comments Based on Gated Attention Neural Networks, *IEEE Access* **6**: 702–710.