INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

GenTex: A Variant of LeakGAN for Text Generation

A dissertation by

Mr Varatharajah Vaseekaran

w1743064 / 2018617

Supervised by

Mr Ragu Sivaramann

Submitted in partial fulfilment of the requirements for the BEng in Software Engineering degree at the University of Westminster.

May 2022

DECLARATION

I hereby certify that this dissertation and all the linked sub-components of it are results of my own doing and none of those have been submitted/presented nor are currently being submitted/presented to any other university or an institution as content for any degree or other qualification program. Extracted facts from credible external sources have been properly cited and given due credit.

Name of the Student: Varatharajah Vaseekaran

Registration Number: 2018617 / w1743064

Signature: Date: 20th of May 2022

ABSTRACT

Text Generation is the task associated with predicting the next word serially and continuously for a given a sequence of text data utilizing language models. A primary use of text generation is data synthesis. Availability of data in machine learning and deep learnings tasks is a constant requirement which can be addressed through generating new data by looking at existing examples.

The evolution of text generation began with statistical models which then progressed to the use of neural networks. However, in both cases, the quality of generated text was limited due to the learning patterns of these models and architectures. These learning patterns should understand context well. To that end, Recurrent Neural Networks (RNN) have gained prominence in addressing contextual understanding for text generation. However, the architectural design of RNN have shown to produce less attractive results in text generation due to their inability to process long text sequences. Though, Long Short-Term Memory addresses this issue, it is less effective under adversarial manipulation. This has turned researchers to utilize GANs for text generation.

The following research focuses on text generation using a modified GANs using COCO Image Captions dataset for training and evaluating. This thesis consists of workings done to modify the architecture of LeakGAN, a GAN-based model for unconditional text generation. The variation of LeakGAN, using Gated Recurrent Units components for the manager and worker modules and modifying the training pattern of the discriminator, displayed improved results: BLEU score of 0.836 successfully beating the benchmark performance of LeakGAN architecture. Thus, this research study has effectively addressed the problem centered around using GANs and models such as RNNs for text generation.

Keywords: Text generation, language models, generative adversarial networks, recurrent neural networks, LSTMs, GRUs.

ACKNOWLEDGEMENT

This year long final year project has been done with the guidance of my friends, family, and mentors, who supported me a lot.

I would like to thank my mentor, Mr. Ragu Sivaraman sir for being immensely helpful throughout my research. A big thanks to Mr. Prasan Yapa sir, for assisting me a lot as I navigated through the literature and helped me narrow down my problem domain. And I am grateful to Mr. Guhananthan Poravi sir, for inspiring me to follow the field of Data Science and AI in our SDGP year.

I would also like to thank my best friend, Pirasanthiny Thayaparan, for being a big help during the research. Thanks to my friends, Anushiya Thevapalan and Gajithra Puvanendran for providing deep insights and support during my thesis period. And I would love to extend my thanks to my Kodemen team, for being in support throughout the final year.

A big thanks to my family, and the IIT staff for assisting us during this final year.

Thank you everyone!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Chapter Overview

The initial chapter of the thesis begins by providing an overview of the research problem addressed by the author of this research study. This is followed by identifying the research gaps, discussing the aim of the research and the challenges that arise when conducting the research. and outlining the scope of the study. The chapter concludes by mentioning the resources that is needed to carry out the study.

1.2 Problem Domain

1.2.1 An Introduction to NLP and Deep Learning

Natural Language Processing (NLP) has been one of the most sought research domains ever since the invention of computers. However, it required many resources and computations to make the machines understand basic human language commands. The breakthrough created by the introduction of neural networks, significantly boosted the research in NLP domain, with machines nearing human-level understanding, thus, enhancing many applications of NLP such as classification of text, generating new text, extracting information from text, etc.

1.2.2 Text Generation

Text Generation in deep learning is the task of synthetically generating text which appear to be indistinguishable from human written text. More specifically, it predicts the next word serially and continuously for a given a sequence of text data. Language models which are created using machine learning (ML) or deep learning (DL) algorithms are used in generating text. The language models are trained to understand the structure and the language constraints of the training data. Through this the language models can observe the inputs in the first t timesteps and generate the output for the t+1 timestep, where each timestep can be a character, a word, or a sentence. Chatbots, paraphrasing software, autocompletion are some applications that utilize text generation and language models.

1.2.3 Generative Adversarial Networks (GANs)

Data availability is one of the key components in ML and DL domain. This has allowed synthetic data generation using adversarial training has gained traction in the research community. Among

them Generative Adversarial Networks (Goodfellow et al., 2014) have gained popularity. Generative Adversarial Networks (GANs) have reinvented the methods of adversarial training and data generation by introducing are two components: (1) the generator which generates synthetic data by trying to mimic the training data and; (2) the discriminator that examines if the generated data seems real or not and sends the information to the generator to improve its synthesis.

1.2.4 Text Generation with GANs

GANs were primarily designed to generate synthetic images. Therefore, the GAN architecture was modified to handle textual data. This was achieved by adding a Gumbel-Softmax differentiation (Jang, Gu and Poole, 2017) to the final layer of the generative model, by utilizing reinforcement learning to update the weights of the models and by modifying training objectives for the GAN.

1.3 Problem Definition

Many text generating applicants use Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTMs) (Hochreiter and Schmidhuber, 1997). However, these models perform poorly in understanding very long sequences of text. Furthermore, due to the gradient vanishing/exploding problem which hampers long term learning, and the inability to process the sequences parallelly, which does not make use of the powerful GPUs for parallel processing. However, recent introduction of Gated Recurrent Units (Chung, Gulcehre and Cho, 2014), known as GRUs, were implemented to be a light weighted version of LSTM, where it performs faster, and better than the LSTM in certain situations (Yang, Yu and Zhou, 2020).

In the past 5 years, transformers (Vaswani et al., 2017) have heavily influenced the NLP domain by replacing the traditional RNN-based models. This is caused by the transformer's ability to solve the issue of understanding very long sequences and enable parallel processing of sequences. However, transformers face issues in adversarial manipulation where perturbations are added to test data to make it different to the distribution of the training data.

Even though significant research has been done on using GANs in the image/video domain, it continues to have a significant influence in the text domain in effectively reducing adversarial manipulation in text generation. However, GANs, require to properly translated to handle textual

data and may encounter mode-collapse where the networks get stuck in local optima causing it to generate the same data over and over.

1.3.1 Problem Statement

Text generation is gaining prominence in research domain is due to requirement of data that can help language modelling tasks for human needs; where GANs architectures success in synthetic image generation has made it a strong candidate in utilizing this for text generation. However, GANs ability to generate good quality text is limited due to the mode-collapse problem and the inherent issues brought by the use of RNN models. This severely limits the capabilities of a machine to generate good quality text which in turn affects is applicability in research domain and use-cases. The research study will modify the GAN architecture for unconditional text generation and tailor an iterative but balanced learning between generator and the discriminator of GANs by utilizing models such as Gated Recurrent Units (GRU) for parallel processing of text sequence and modifying training pattern of discriminator; which is then tested on a benchmark dataset to prove the success of the methods employed.

1.4 Research Motivation

Text Generation is a challenging and formidable research problem, with plenty of literature available, and room for improvement. Personally, taking on such a research problem would help one to enhance the knowledge of many state-of-the-art methods and technologies, such as Language Models, RNNs, LSTMs, GRUs, Transformers, RL, GANs, etc. Also, managing to provide an ample solution for some of the research gaps detected can be fruitful for future opportunities in Text Generation and NLP: on generating quality text, stable model training, formulating new algorithms, and modifying existing architectures. The research would also assist in guiding future enthusiasts on conducting their own research on the same problem domain.

1.5 Existing Work

Literature suggests that several research works have been conducted in the field of Text Generation in the last decade using techniques such as RNNs, attention models, GANs, and Reinforcement Learning. This section would provide a brief insight into the existing text generation methods that are considered important for conducting the research.

Citation	Brief Description	Improvements	Limitations
(Guo et al., 2017)	GAN-based text generation utilizes RL and a leaking feature, where the discriminator leaks information to the generator.	Outperforms previous state-of-the-art models; better in generating long sequences	Uses RNN-based components; training time is high
(Wu, Li and Yu, 2021)	Combines large-scaled pre-trained transformers (GPT-2 and RoBERTa) with RL.	Stable training, constant rewards were provided.	Pre-trained model sizes are large, computationally expensive
(Lin et al., 2018)	Uses a rank-based generation that evaluates and ranks human- and machine-written sequences by using a relevance score like cosine similarity.	Removed the need of training the discriminator	Not well for long sequences; did not achieve good results in benchmarking
(Fedus, Goodfellow and Dai, 2018)	Uses a Seq2Seq generator and is trained using policy gradients via REINFORCE algorithm	Reduces the problem of mode collapse, helps to reduce the risk of discriminator outperforming the generator	Uses RNN-based components.

(Yu et al., 2017)	Uses REINFORCE algorithm and Monte-Carlo search to train the model.	First RL-based GAN for text generation, a hallmark for other GAN-based generation	Lacked generalization power; generated text with minimum diversity

Table 1 - A summary of important works

1.6 Research Gap

Most of the state-of-the-art text generation methods use RNNs and/or variants of RNNs as components in their generator architecture. Though RNNs provide an ideal method for generating text, they face some challenges such as adversarial manipulation, inability to process longer sequences, only being able to process data in a sequential manner, etc. There exists a research gap in text generation method which replaces the RNN-based components in the generator with different memory components or models could potentially help to resolve the above challenges. This research gap can be stated as a performance (the text quality and diversity) gap.

GAN-based structures for text generation are susceptible to mode-collapse, and there is room for improvement in modifying the discriminator of the GAN, as it can extract and classify the real/fake data better to provide meaningful information to the generator. This research gap can be defined as a performance gap.

Unlike major languages (i.e., English, French, Chinese, etc.), low-resource languages have a lack data. Thus, research on text generation in low-resource languages are limited. Existing studies have generated poor quality text for low-resource languages which indicates that there is a research gap in generating quality text for this specific type of textual data.

1.7 Contribution to the body of knowledge

1. Technical Contribution

RNN-based architectures have established a benchmark in many text generation datasets, but as discussed earlier, RNNs have a lot of shortcomings. The research would replace the RNN-based components with different model components in the model architecture.

2. Domain Contribution

   In most text generation research, there is room for improvement for the generated text's quality and diversity. Therefore, the following research would provide the contribution of generating better text. Additionally, the language that is mainly targeted would be English and utilizing the datasets, the following research would be beneficial in implementing a better text generation method for the English language. If a low-resource language is chosen for text generation, then a dataset of the chosen low-resource language would be contributed along with benchmark scores for the findings obtained from the research.

1.8 Research Challenge

- Computational Complexity: Training deep learning models for generating text require computational resources; therefore, successfully training and building models would be challenging from the resource perspective.
- Technological knowledge: For this research, a lot of trending deep learning topics are set to be researched and learned, such as GANs, language models, reinforcement learning, and transformers. Therefore, obtaining knowledge of the methods can be considered a challenge.
- Knowledge on tools: Obtaining proper knowledge and experience in utilising tools, such as Python, PyTorch, TensorFlow, NLTK, and other necessary tools needed to conduct the research.
- Implementation and Evaluation: The initial plan of the research is to replace the RNN-based components in state-of-the-art research in generating text to transformer components. This would be challenging as RNNs and transformers have different architectures and different input and output formats. Also, creating a stable benchmarking pipeline to evaluate the results would provide challenges.

The research study has arduous challenges. However, the author's interest in the research domain and as well as the technologies used can help to successfully bridge past the challenges. Furthermore, as the challenges clearly depict, the following research can be extended as postgraduate research as well, where more focus can be provided to address the issues such as generated text's quality, low-resource scenarios, faster training, etc.

1.9 Research Questions

Research Question 1: What are the existing methods in the field of Text Generation?

Research Question 2: How can Transformer-based components improve the existing state-of-the-art text generation methods?

Research Question 3: What are the existing text evaluation methods and how to use them?

1.10 Research Aim

The aim of the research is to analyze, design, implement, and evaluate a text generation model that could successfully take a dataset of text sentences as input to generate meaningful and quality sentences.

To further elaborate on the aim, the research project would provide a solution that takes starter words of text as input, and generates a paragraph of words, until the maximum number of words or convergence is reached. The aim would be to clearly evaluate the existing state-of-the-art implementations and research in text generation and build a new algorithm/model that would improve on the previous works.

Over the course of the research, the needed knowledge for successfully completing the project would be learned, the needed components for the project would be built, and the results would be evaluated in order to determine whether the research hypothesis has been successfully achieved.

1.11 Research Objectives

Research Objectives	Explanation	Learning Outcome(s)
Problem Identification	• Identifying a research gap in text generation methods • Identifying a proper solution is needed to solve the problem: text generation • Identifying the ethical and legal issues around the problem if any issues are present.	LO4, LO6, LO2
Literature Review	• Carrying out an in-depth and critical analysis in the existing works in the text generation domain • Carrying out an in-depth analysis in the evaluation methods for text generation techniques	LO4, LO5
Data Gathering and Analysis	• Carrying out an in-depth analysis for the requirements that are needed to conduct the research • Carrying out in-depth analysis by getting insights from technological and domain experts to implement the research successfully • Identifying the functional and non-functional requirements of the research project	LO2, LO3, LO4
Research Design	• Designing a flow diagram for how the overall system would work in generating text • Designing the model components for the generation components. • Designing an evaluation process to test the generated outputs	LO1, LO2, LO3
Implementation	• Developing an initial prototype that would successfully perform text generation	LO1, LO7

	• Developing an evaluation process that would provide insights on the output's quality.	
Testing and Evaluation	• Evaluating the system with new data that does not belong to the training set • Evaluating the system with the insights from technological and domain experts • Testing the functional and non-functional requirements • Evaluating the system by using benchmark datasets • Writing a detailed report on the conducted research	LO7, LO8

Table 2 - Research objectives

1.12 Project Scope

Based on the research objectives that are defined above and the review of existing literature, the scope is as follows: the main intention of the following research project is to design, implement, and evaluate an improved Text Generation model that would generate better text, with shorter training period and better quality.

1.12.1 In scope

The scope that is covered in this research project is as follows:

- Initial model creation: Building an initial model that would successfully generate text based on starter words.
- Model improvements: Iterating upon the initial solution by modifying it to improve the quality of the text generated.
- Summary evaluation: Researching and obtaining a good evaluation metric that would provide proper feedback on the generated text.

1.12.2 Out scope

- The proposed system does not focus on other NLP tasks; only for text generation.

- The proposed system would support only the language that it has been trained on, and not any other languages (that is if the model is trained with English, it would only support English, and not other languages like Tamil, Sinhala, French, etc.)

1.12.3 Prototype Feature Diagram

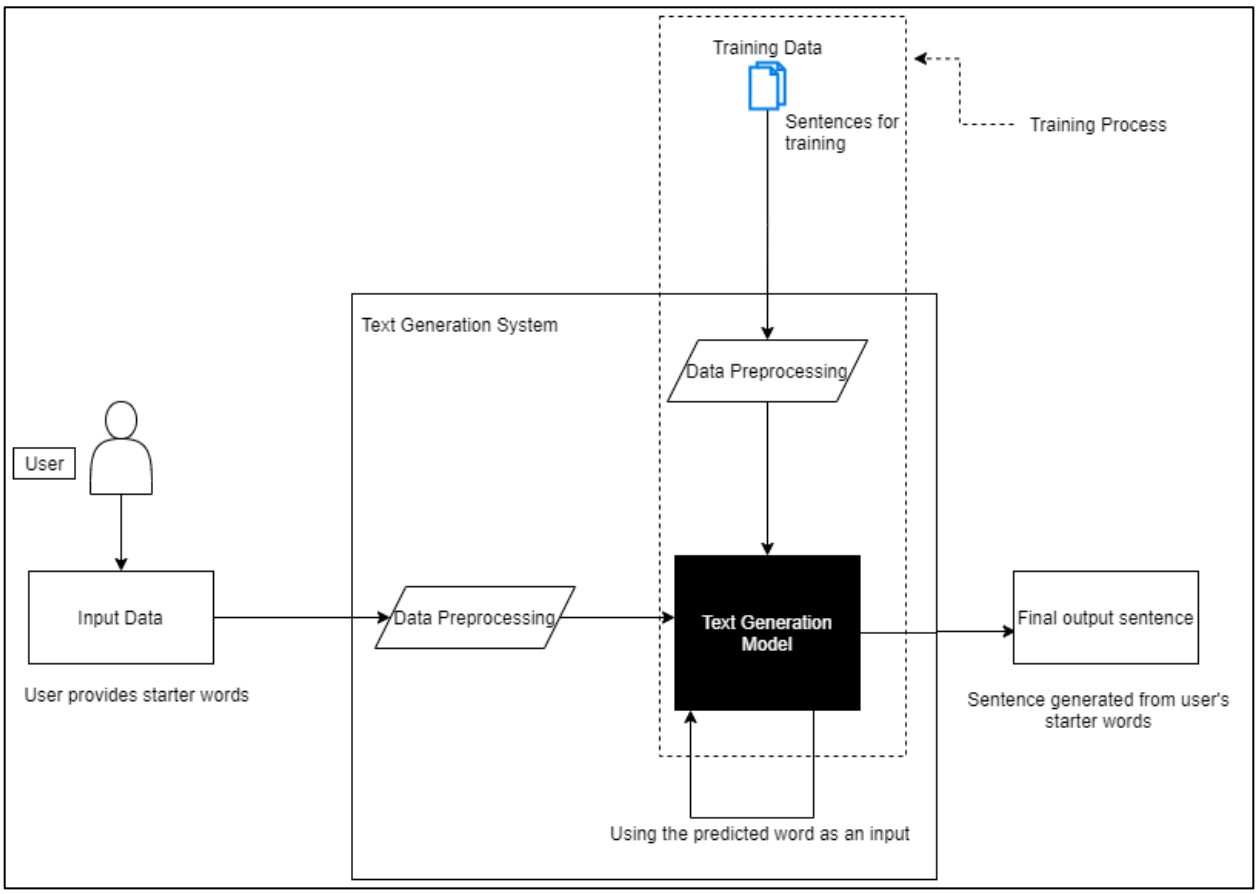


Figure 1 - Prototype diagram

1.13 Resource Requirements

1.13.1 Hardware Resource Requirements

Requirement	Resource(s)	Reason

Processor	Intel Core i7/i9 processor **or** AMD Ryzen processor	For performing resource-intensive tasks (model calculations, pre-processing, etc.)
RAM (Memory)	12/16 GB	For storing model variables and datasets needed for processing
Disk Space	More than 20GB	Mainly for storing datasets and model files
Graphics (GPU)	NVIDIA RTX (with/without Ti) with at least 6GB Video Memory	For faster computations of Deep Learning algorithms

Table 3 - Hardware Resource Requirements

1.13.2 Software Resource Requirements

Requirement	Resource(s)	Reason
Operating System	macOS/ Windows 10/Ubuntu	for viewing and handling the process of building algorithms
Programming Language	Python/R	For coding the needed algorithms and models (Mainly Python would be preferred for building Deep Learning algorithms)
ML libraries	Sci-kit Learn/Catboost/XGBoost/LightGBM/mlr	For building ML models in case if ML is needed
DL libraries	PyTorch/TensorFlow	For building deep learning models

NLP libraries	NLTK, Spacy	For handling textual data and preprocessing
Visualization libraries	Seaborn/Matplotlib/Tensorboard	For visualizing the data and model progress
Development environments	VSCode/PyCharm/Jupyter Notebooks	For designing and building code
Online Development Environments	Google Colab/Kaggle Notebook/AWS Sagemaker/Gradient Paperspace	For designing and building code with GPU available
UI tools	Streamlit/ React-based libraries	For designing a simple frontend
Reference manager	Zotero/ Mendeley	For managing the existing research literature and artefacts
Word editor	MS Word/Google Docs/Pages	For documenting and reporting
Version manager	GitHub/ Bitbucket	For storing and versioning the code
Backup manager	Google Drive/Dropbox	For backing up important files related to research

Table 4 - Software Resource Requirements

1.13.3 Data Requirements

- Natural Language Datasets:
    - COCO Captions - From Microsoft, images with captions. The captions can be separately used for text generation.

    - EMNLP WMT News - News for the translation use case, but data from specific language can be used for text generation.

- Other Dataset Sources (For searching additional data):
    - Kaggle
    - UCI Repository
    - Google Dataset
    - DataWorld
    - *Other Private Sources*
- Deep Learning models: HuggingFace/TensorFlow Hub/Torchvision.Models - Pretrained models can be obtained for faster training.

1.13.4 Skill Requirement

- Handling Textual Data: Pre-processing, tokenizing, etc.
- Building Deep Learning models: Needed for building models to handle text data, especially RNN-based DL.
- Using DL Frameworks: Expertly using PyTorch/TensorFlow requires more knowledge and practice.
- Building GANs: Knowledge of GANs is required to successfully design and build models
- Building Transformers: Knowledge of Transformer architectures is needed
- Reinforcement Learning: Knowledge of RL is needed
- Search strategies: Knowledge of search strategies for text generation is important
- Designing benchmarking: Knowledge on evaluating and benchmarking the models is crucial to successfully evaluate the new algorithm.

1.14 Chapter Summary

A brief introduction to the research was provided, as the domain of the problem and the research gap were discussed. The objectives of the research were also stated, and basic requirements needed for the project was also explained.

CHAPTER 2: LITERATURE REVIEW

2.1 Chapter Overview

The literature review would provide a detailed explanation of the research topic that has been selected, Text Generation, and would provide a critical analysis of the existing works, tools, evaluation approaches, and benchmarks that would assist in presenting the problem domain and the research gap.

2.2 Concept Map

A map that clearly displays the domain around text generation, where it explains the problem of text generation, the existing ways on how to solve it, the improvements that can be made, the evaluation metrics, etc. The map can be found in the appendix.

2.3 Review of the Text Generation Domain

2.3.1 Natural Language Processing

A simple definition for Natural Language Processing (NLP) can be defined as teaching machines to process and understand human languages. Ever since the invention of the computer, active research has been conducted to make machines understand natural languages. With the advent of the internet era, along with the rise of neural networks, NLP has undergone revolutionary research and development in the past two decades, which paved the way for human-like machines. NLP has influenced many industrial as well as research domains in the past decades: sentiment analysis for reviews and social media posts, building up language models that would understand a language and then generate new sentences, text-to-speech and speech-to-text systems, translating one language to another efficiently, text summarization, and chatbots are some of the many use-cases that NLP has helped to revolutionize.

2.3.2 Text Generation

Language Models are the main components that assist in generating text. A simple explanation of text generation can be stated as to when providing textual data as input, a newer set of textual words would be generated. To train a text generation model (or a Language Model), the model would be fed with training textual data, so the language model can capture the grammatic, syntactic and semantic rules. Then based on what the model has learned, it can predict the next words when a set of words is provided as input. That is, the model can be used to predict t+1 time steps, based on t time steps, where one time step can represent a character, a word, or even a sentence.

While Text Generation is used in various applications currently, it is a heavily researched field that works on improving the quality of the generated text. Autocomplete is one such use case of text generation which is used in many emails, text, and word processing applications, where it predicts the text that the user might prefer when the user is typing. Question Answering is a feature that is observed in the present-day chatbots and is powered by text generation, where the model learns to answer questions asked by the user, and this paves the way to human-like answering from machines. Text summarization is another use case of text generation, where long paragraphs of text can be summarized into shorter sentences. With the advancement in the deep learning field, abstractive summarization, which is known as paraphrasing, became an interesting topic in research, whereas extractive text summarization, which uses preprocessing steps to shorten the paragraph, by removing and editing the words was popular earlier.

2.3.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) have been a trending research domain in computer science for the past years. It introduced and reinforced many opportunities in computer vision and deep learning, and it has also successfully transitioned to handle sequential data. The main idea of a GAN is to generate a distribution from some random noise, where the generated distribution would match the distribution of the real data. GAN has two networks for achieving its objective: a generator, which would transform a random noise distribution into a distribution that represents the real data, and a discriminator, which would predict whether the generated data is closer to the real data or not. The discriminator would feed information to the generator, so the generator can

learn to generate better data that resembles the real data. So, over time, both generator and discriminator try to beat each other, and thus, it's called an adversarial network.

GANs were primarily developed for image-based data generation. When using Recurrent Neural Network architectures as GAN components, it is very difficult to differentiate the Generator's output, as for RNNs, the output is obtained by sampling one-hot categorical variables from a Softmax distribution. That is, GANs were designed to handle continuous information, but not discrete data, so they are not suitable when applied with sequences of text. Therefore, in order to use GANs for textual data, the architectures are modified using 1) Gumbel-Softmax Differentiation; 2) Reinforcement Learning; or 3) modifying the training objectives of the GAN.

Gumbel-Softmax is a continuous distribution that can be used to obtain approximate samples from categorical distributions, and the continuous distribution allows the gradients of the network to be differentiable, so the network can be updated during backpropagation. Using stochastic gradient policy updates in another way to tackle the generator's differentiation problem when handling textual data, is using reinforcement learning (RL). The main idea here is to obtain reward signals from the discriminator (obtained during fake/real classifications) and send the reward signals to the generator, for the generator to learn, just the way a reinforcement algorithm would perform (e.g. REINFORCE algorithm). And the final way, which is an alternative to Gumbel-Softmax and RL-based approaches, is modifying the training objectives of the GAN, where different approaches such as Maximum Likelihood Estimation(MLE)-like method, using AutoEncoders, using distance-based functions are utilized.

2.4 Existing Work

2.4.1 GANs with Gumbel-Softmax Layer

As GANs are trained to handle continuous data (like pixel values), some modifications must be done to make GANs handle discrete data (textual data). One such way is using a Gumbel-Softmax layer(Jang, Gu and Poole, 2017), which transforms a discrete data distribution into a continuous one. Few existing works have successfully utilized the Gumbel-Softmax layer in GANs to build language models and generate texts.

The earliest implementation of the Gumbel-Softmax layer with GAN is the GS-GAN, known as Gumbel-Softmax GAN(Kusner and Hernández-Lobato, 2016), which is a GAN based on recurrent neural networks, with Gumbel-Softmax final layer, that generates sequences of discrete elements. This research was conducted using five thousand training samples that have a maximum length of 12 characters. This work proved that Gumbel-Softmax can be used in GANs for text generation, however, the results obtained from the generated texts were not comparable to the state-of-the-art models for text generation.

Relational GAN (RelGAN)(Nie, Narodytska and Patel, 2018) uses a relational memory-based generator with the Gumbel-Softmax output layer, and a multi-embedded representation discriminator and experiments were conducted with synthetic data (generated from an Oracle LSTM(Yu et al., 2017)) and real datasets (such as COCO image captions(Chen et al., 2015), WMT news(Bojar et al., 2017)), and RelGAN produced results that are comparable to the state-of-the-art models (MLE, SeqGAN, RankGAN, LeakGAN). The relational memory helps the generator to capture long term dependencies in sequences, thanks to the self-attention layer. However, RelGAN requires proper pre-training to generate meaningful text, and this increases the computational cost of training the model.

Meta Cooperative Training Paradigm GAN (Meta-CoTGAN)(Yin et al., 2020) attempts to solve the mode-collapsing problem that GANs face when they are trained, by using a cooperative training language model and meta optimization. Even though mode collapse was significantly reduced, the high computational cost was present, as another language model has to be trained.

2.4.2 GANs with Reinforcement Learning

Most of the Gumbel-Softmax based GAN models for text generation require pre-training before adversarial training and also rely on traditional GAN objectives, therefore those works are prone to premature collapse and an unbalanced equilibrium between the discriminator and the generator. An alternative approach is using stochastic gradient policy updates to bypass the generator's differentiation problem, that is obtaining reward signals from the discriminator and passing those

signals to the generator: that is a Reinforcement Learning method, such as the REINFORCE algorithm(Williams, 1992).

Sequence GAN (SeqGAN)(Yu et al., 2017) is the first RL-based research, and this uses the REINFORCE algorithm(Williams, 1992) and Monte Carlo searches to transfer the discriminator's gradients to the generator. SeqGAN is comparable only to MLE-based text generation models, but SeqGAN established a benchmark model and a starter model for the future RL-based research in text generation. Even though SeqGAN was able to successfully generate text from adversarial training, SeqGAN had issues in generalization as it generated text with less diversity.

A modified version of SeqGAN (VGAN)(Wang, Qin and Wan, 2018) was implemented, which uses additional features obtained from a Variational AutoEncoder(Kingma and Welling, 2013) to represent text; the high-level random variables help to model the textual data's variability and this assists the RNN-based GAN components in learning the well-structured data. The same gradient policy used in SeqGAN is used here. The VGAN outperforms RNN-based Language models and SeqGAN in three public benchmark datasets.

RankGAN(Lin et al., 2018) uses a ranking-based architecture to generate text. It evaluates and ranks real and generated sequences over a reference group, instead of having a discriminator to predict whether the sequences are real or not. This allows the model to compute the expectations given different references, which are sampled across the reference space, and to obtain such a value, a relevance score is formulated, which uses cosine similarity followed by a softmax activation, and this calculates a collective ranking score for an input sequence.

Leaked Information GAN (LeakGAN)(Guo et al., 2018) is one of the state-of-the-art GANs for generating text, and it works well when generating long sequences. The discriminator leaks high-level features (that are used for classification) to the generator, so this additional information helps the generator to learn better during training. The generator has 2 important components: the MANAGER builds a feature vector from the leaked information, and the WORKER works to generate text by using the built feature vector. Both MANAGER and WORKER are RNN-based components, and the REINFORCE algorithm(Williams, 1992) is used to train LeakGAN. Also,

this research introduces Bootstrapped Rescaled Activation, where the reward values are rescaled to deal with the vanishing gradient problem(Pascanu, Mikolov and Bengio, 2013) in long sequences.

Self Adversarial Training (SAL) (Zhou et al., 2020) compares the quality of the text by using a comparative discriminator, where throughout the training step, the generator is rewarded when the generated sequence is better than the previous one. This helps to reduce mode collapse and reward sparsity.

Feature Guiding GAN (FGGAN) (Yang et al., 2020) uses a CNN component as a feature guidance module, where a feature vector can be obtained from the real and generated sequence, and that vector is used as a kind of reward signal as well. This research also established certain semantic rules to remove garbage and illogical word tokens, in order to improve the quality of the generated text.

2.4.3 GANs and modified training objectives

Maximum-Likelihood Augmented GAN (MaliGAN) (Che et al., 2017) uses a novel objective, that is using normalized Maximum Likelihood and importance sampling, and this makes the training more stable. Another important discovery in this method is that the discriminators are able to learn if the generator has learned too much noise due to the variation reduction techniques.

JSD-GAN, known as Jensen-Shannon Divergence GAN (Li et al., 2019), uses an alternative to the min-max optimization, that is the Jensen-Shannon divergence (Nielsen, 2020), to optimize the distribution of the generator and the training data. It is an RL-free algorithm, but this can be only applied to tasks with explicit representations.

TextKD-GAN, that is Text Knowledge Distillation GAN, introduces continuous text representation, that would replace the traditional one-hot encoding of texts. This model uses an AutoEncoder and a teacher-student model, where the data is reconstructed after it is fed through the AutoEncoder. The generator is well trained here, and it becomes a hard challenge for the discriminator to work.

2.5 Review of Problem-Solving Approaches

Language modelling and generating text can be solved with various techniques: machine learning models, deep neural networks, and statistical models. Following are some of the important methods that can assist in generating text.

Recurrent Neural Networks (RNNs) (Hopfield, 1982) are an important discovery in the field of Deep Learning, as they enabled neural networks to process sequential data efficiently and accurately, compared to the standard feed-forward neural networks. The speciality of the network is that the current timestep in the model is fed to the next timestep, which enables the model to capture information across a sequence. RNNs are used extensively in many text-related applications, such as classification, generation, information retrieval, etc. However, RNNs face a major issue; they cannot process long sequential data well and is prone to gradient vanishing/exploding problems (Pascanu, Mikolov and Bengio, 2013).

Long Short Term Memory (LSTM) (Hochreiter, Rovelli and Winckler, 1997) architecture was introduced to bridge the issues faced by the RNN models. LSTMs are equipped with memory gates, where important information across the sequence is conserved. This enables the model to process longer sequences and provides an antidote to the gradient vanishing and exploding issues. The cell state in LSTM is not exposed to other units in the network, and it has separate input and forget gates.

Gated Recurrent Unit (GRU) (Chung et al., 2014) is another replacement for the traditional RNN based models, as they are equipped with memory gates to prevent information loss in long sequences and to prevent gradient vanishing and exploding issues. The GRU has only one gate, the reset gate, where the decision occurs regarding which information is crucial to remember. The cell state in the GRU is exposed to the other units in the network.

Convolutional Neural Networks (CNNs) (LeCun et al., 1998) are a milestone in the field of image processing, as they established state-of-the-art results in many image-related tasks. The main strength of CNN is to build a feature vector that can accurately represent the important

aspects of the real data, in a fraction of the original size and enables the use of the parallelization power of Graphical Processing Units. CNNs are successfully used in text data as well, as by using one-dimensional convolutional operators (Kiranyaz et al., 2019), it is possible to obtain feature vectors for text, which would aid in tackling text-related tasks.

Transformers (Vaswani et al., 2017) are a relatively new architecture which is trending in the field of deep learning. With an encoder-decoder architecture, and attention network, where the focus is given to important data, transformers are outperforming many recurrent-based models. And, transformers are parallelizable, as the data can be fed in chunks, and not entirely as a sequence. Also, there are encoder-only and decoder-only transformers, such as BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019), and T5 (Raffel et al., 2019), which perform impressively in many image-related tasks. However, transformers are a bit complex to understand and complicated to integrate with other models in different architectures.

2.6 Review on Benchmarking

There are plenty of datasets that can be used for text generation, ranging from low-resource languages to famous languages. However, to produce results that can be compared with existing works and state-of-the-art models, benchmark datasets are preferred, and such types of datasets are considered for the research. Following are some of the benchmarking datasets that are considered to conduct the research.

The **COCO Image Captions** dataset (Chen et al., 2015) is created by Microsoft, where around a million images and their captions can be obtained. In order to conduct the text generation task, the human-generated captions are obtained and are used to build language models and text generation models.

WMT News from EMNLP2017 (Bojar et al., 2017) is a dataset that is primarily used for language translation but can also be used for generating text. Sentences from one language are selected for generation, and this dataset has been acting as a benchmark for many language models.

2.7 Review on Evaluation Techniques

Evaluating an NLP research requires unique metrics that differ from the traditional data science evaluations. As the machine creates synthetic data by attempting to understand the human language, it is crucial to carefully evaluate the data with respect to human-created data. Therefore, algorithms were implemented to facilitate such evaluations and to be used for future benchmarking purposes.

Bilingual Evaluation Understudy (Papineni et al., 2002), which is known as the **BLEU** score, evaluates the machine-generated text with respect to the ground truth, which is the human-generated text. The co-occurrence of n-grams (Nagalavi and Hanumanthappa, 2016) between the generated text and the ground truth text are compared, which computes a precision score, where the maximum value would be 1.0, which indicates the generated sequence is identical to the ground truth sequence. Also, a penalization factor is used to compute the BLEU score, as short sentences can inaccurately display high scores.

Negative Log-Likelihood (NLL) (Myung, 2003) is usually used to maximize the probability of correct predictions in ML algorithms, and in the task of generating text, NLL is to ensure that the model is generating t+1 tokens, given t tokens. Therefore, NLL is used to monitor the stability during generating text. The metric ranges from 0 to positive infinity, where a value closer to zero indicates that the model is working well and generating text as intended.

2.8 Review on Existing Works

Based on the existing works, and the evaluation metrics comparing each work, the author discovered that the LeakGAN architecture is suitable for extensive research and would be the core component of the research. The research would focus on modifying the LeakGAN architecture in an attempt to improve the existing scores that were obtained by the LeakGAN model on the benchmark datasets.

The following are the results mentioned in the LeakGAN paper when the tests were conducted on the selected two benchmark datasets.

Table 2: BLEU scores performance on EMNLP2017 WMT.

Method	SeqGAN	RankGAN	LeakGAN	p-value
BLEU-2	0.8590	0.778	**0.956**	$< 10^{-6}$
BLEU-3	0.6015	0.478	**0.819**	$< 10^{-6}$
BLEU-4	0.4541	0.411	**0.627**	$< 10^{-6}$
BLEU-5	0.4498	0.463	**0.498**	$< 10^{-6}$

Figure 2 - The results from the LeakGAN (Guo et al., 2018) paper for EMNLP2017 dataset

Table 3: BLEU scores on COCO Image Captions.

Method	SeqGAN	RankGAN	LeakGAN	p-value
BLEU-2	0.831	0.850	**0.950**	$< 10^{-6}$
BLEU-3	0.642	0.672	**0.880**	$< 10^{-6}$
BLEU-4	0.521	0.557	**0.778**	$< 10^{-6}$
BLEU-5	0.427	0.544	**0.686**	$< 10^{-6}$

Figure 3 - The results from the LeakGAN (Guo et al., 2018) paper for COCO Image Captions dataset

However, as the results were obtained in 2017, and due to an update in the NLTK package, which is used to obtain the BLEU score for the generated texts, the above results could not be reproduced by other publications. The following are the updated result obtained from training GAN-based models for generating text, and the performance of the LeakGAN for the selected two benchmark datasets can be observed.

Table 2
BLEU-2 benchmark regarding architectures that use such a metric to evaluate the four most common datasets.

Architecture	Amazon Review	Chinese Poems	COCO Captions	WMT News
MLE	-	0.667	0.781	0.768
RNNLM	0.848	-	-	-
RelGAN	-	-	0.849	0.881
Meta-CoTGAN	-	-	0.858	0.882
SeqGAN	0.856	0.739	0.745	0.777
VGAN	0.868	-	-	-
RankGAN	-	0.812	0.743	0.727
LeakGAN	-	**0.881**	0.746	0.826
IRL	-	-	0.829	-
BFGAN	**0.920**	-	-	-
SAL	-	-	0.785	0.788
FGGAN	-	-	0.773	-
MaliGAN	-	0.741	-	-
JSD-GAN	-	0.536	**0.894**	**0.943**

Figure 4 - Results obtained from (de Rosa and Papa, 2021)

Figure 5 - Results obtained from (de Rosa and Papa, 2021)

The updated metrics show that LeakGAN has performed poorly, compared to the results obtained in the paper and also compared to other models. Therefore, the research would focus on modifying the architecture and the training methods of the LeakGAN and attempting to improve the results.

Recurrent models would be used as architecture components to modify the core component of the research. Initially, the author had planned on using transformer-based models, but due to their complexity, recurrent models were chosen.

BLEU and NLL would be the loss functions selected to evaluate the model, as the existing research was done mainly using the two metrics. The research would be trained with the benchmark datasets

that were discussed, as it would be comparable to the base model and also would be highly useful when benchmarking with other existing works.

2.9 Chapter Summary

The chapter focused on the existing literature surrounding the problem domain. The problem domain was explained, and then, some of the important existing works were discussed. Then, a review was done of the existing tools, metrics and benchmarking datasets, and finally, a review of the existing works was done, in order for the author to extract insights to conduct the research.

CHAPTER 3: METHODOLOGY

3.1 Chapter Overview

The following chapter gives a detailed report based on how the research is conducted. The research methodology and the development methodology would be explained. The deliverables that would be generated from the research would be explained, and the chapter would conclude with the risks and mitigations faced during the research.

3.2 Research Methodology

The research methodology is based on the Saunders's Research Onion model and is stated in the following table.

Research Philosophy	Out of the four research philosophies (Pragmatism(Morgan, 2014), Positivism(Park, Konge and Artino, 2020), Realism(House, 1991), and Interpretivism(Chowdhury, 2014)), **Pragmatism** philosophy is chosen. Pragmatism is chosen as the following research needs actions and experiments to successfully come up with a solution, where the results of the actions would decide on what changes to be done to build the solution
Research Approach	Out of the three research approaches (inductive(Liu, 2016), deductive(Hyde, 2000), and abductive(Dubois and Gadde, 2002)), the **deductive** approach is chosen. Deductive approach is chosen as the research would be finding the best possible solution after evaluating the proposed theories with the available data.
Research Strategy	Out of the available research strategies (experiment, survey, archival research, case study, ethnography, action research, grounded theory, narrative inquiry), the **experiment** is chosen as the primary strategy, and **surveys** and **interviews** also would be used in the testing and feedback stages. Experiments are chosen for text generation, based on a specific independent input, we would be obtaining a dependent output that would be evaluated to check whether the generated text is good or not.
Research Choice	Out of the available research choices (mono, mixed, and multi), **mixed-method** is chosen as the research choice. As explained in the Research Strategy, the following research uses both qualitative methods (surveys and interviews) and quantitative methods (experiments and performance based on numeric values) are used.

Time Zone	Out of the two time-zone studies (cross-sectional and longitudinal), **the cross-sectional** study is chosen. Cross-sectional is ideal as it allows the selection of a new set of people to evaluate the research at a specific point in time. Having the same set of individuals for evaluation throughout the project at different points in time is infeasible, therefore, cross-sectional is better.

Table 5 - Research Methodology

3.3 Development Methodology

Life Cycle Model: Iterative model

The proposed research project requires adding new features and improving the research performance after initially implementing a base working prototype. So, the iterative model is the best choice for the life cycle model.

Starting with an initial model/project, each iteration would focus on adding or modifying components with the aim of improving the performance of the intended system. The modified version of the model/project would be evaluated based on certain metrics (that would be explained later in the proposal), and based on the results, the next iteration would be performed.

Design Methodology: Structured Systems Analysis and Design Method (SSADM)

SSADM allows to break up the development into smaller modules and steps, so each module can focus on improving the output of the research. SSADM also is good for managing potential risks that can be faced during development, making the project manageable with maintainable codes and components, and also more likely to meet the requirements that are set.

Evaluation Methodology

Text generated from computers can be quite tricky: if it's a well-trained model, then the model would generate human-like sentences, or else, it would be messy and wouldn't make sense. Generated text can be evaluated using normal inspection as well as some automated evaluation. This research would focus on automatically evaluating the generated text using established text evaluation metrics, such as BLEU, ROUGE, negative log-likelihood, perplexity, LSA, METEOR, and TER.

Benchmarking

The aim of the research is to design and implement a text generation method that can exceed the performance of the state-of-the-art implementations and to successfully prove the research hypothesis, benchmark datasets are required. The following research would use two benchmarking datasets: COCO captions dataset from Microsoft, and the WMT news dataset.

3.4 Project Management Methodology

Personal Scrum would be chosen as the project management methodology. Scrum supports Agile project management, as well as the iterative life cycle model, as each sprint can focus on delivering a finished model on generating text, and each iteration can improve on the previous implementation. The short development phases in Scrum allow faster evaluation and modification of the research system. After each sprint, evaluation metrics and professionals would assist in evaluating the model/algorithm and for further improvements.

3.4.1 Project Plan (Gantt Chart)

The Gantt chart, that features the project breakdown and the time scheduled for the tasks, can be found in the appendix.

3.4.2 Deliverables

Specified and discussed in detail; the associated risks are evaluated and possibly deliverables are prioritized accordingly.

Deliverables	Date
Project Initiation Document The project proposal for the research	11th November 2021
Literature Review Document A Critical Review and Analysis of the existing literature and methods regarding the chosen research	18th November 2021
Software Requirement Specifications The document which specifies the needed requirements for successfully completing the project	25th November 2021
Proof of Concept/Initial Prototype A basic working prototype for establishing that the research is progressing	6th December 2021
Final Prototype A final working prototype that successfully establishes that the research is completed	March-April 2022
Thesis The final report contains all the workings and processes that had been done to complete the research project	20th May 2022
Review Research paper A publishable review research paper that contains the existing works and critical analysis of those works.	January-May 2022
Final Research paper	May 2022

A publishable research paper that introduces the new method taken to solve the research problem	
Publish Code/Model Code repo and model of solving the research problem would be published that can be accessed by everyone.	May 2022

Table 6- Deliverables

3.4.3 Risks and Mitigation

Risk Item	Severity	Frequency	Mitigation Plan
Mode collapse during training	4	4	Interleaved training was utilized, models were saved at each checkpoint
Obtaining poor evaluation scores	5	4	Using powerful models, using pre-trained models, using different pre-processing techniques
Slower training	3	4	Using GPUs, using cloud notebooks
Obtaining data for training	2	2	Scraping from the web, using data from external sources like Kaggle
Generating less diverse text	4	4	Using pre-trained models, modifying training parameters, using different textual data
Data deletion/corruption	5	3	Using cloud backups, external hard drives

Table 7 - Risks and Mitigation

3.5 Chapter Summary

In this chapter, the methodology used for the research was discussed. Development methodology was also discussed extensively, as the design, evaluation and benchmarking were discussed. Also, the deliverables of this research were discussed, and the risks faced during the research and the way on how they were solved were discussed.

CHAPTER 4: SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Chapter Overview

This chapter focuses on the requirements gathering process for building up the research project. Initially, the stakeholder and domain analysis are depicted with the Stakeholder Onion Model and the Rich Picture diagram. Then, the methods of how the requirements elicitation was done are explained and the findings from the methods are discussed. Finally, the chapter concludes by discussing the context diagram, use-case diagram, and the requirements of the project.

4.2 Rich Picture

The rich picture diagram visually represents how the proposed research system interacts with the environment and visualizes how the final product can be used.

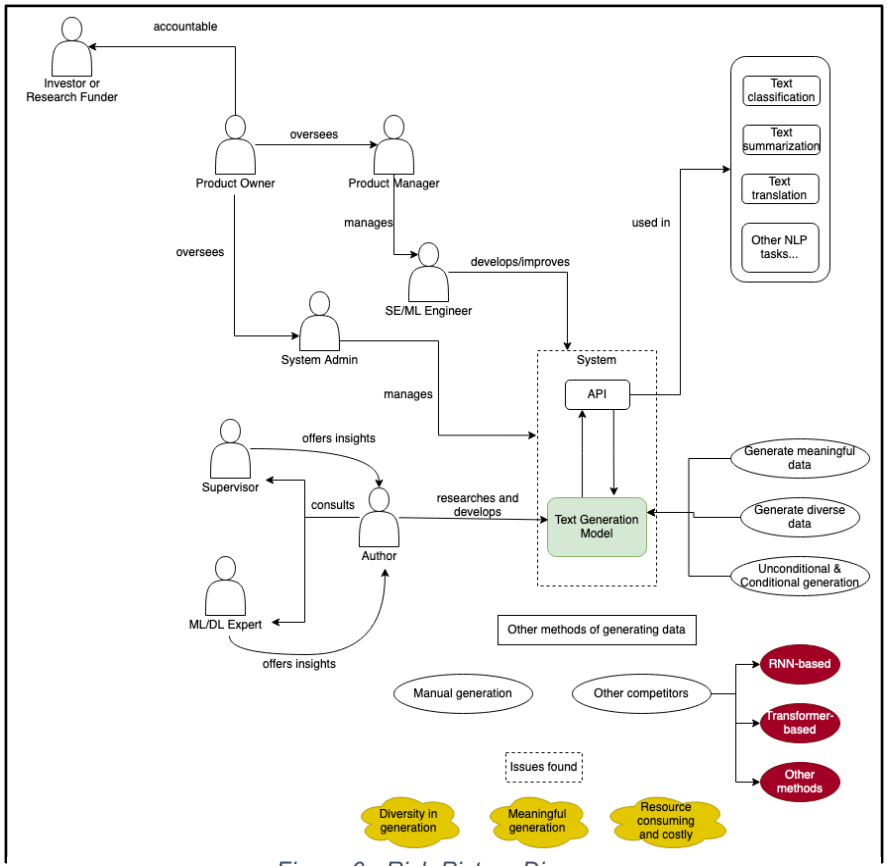


Figure 6 - Rich Picture Diagram

Figure 7 - Rich Picture Diagram

4.3 Stakeholder Analysis

The stakeholder analysis identifies the different personalities (stakeholders) that can influence the system. The Stakeholder Onion model provides a visual representation of how the stakeholders influence the system, and the Stakeholder viewpoints provide a detailed description of the role and the duties of each stakeholder.

4.3.1 Onion Model

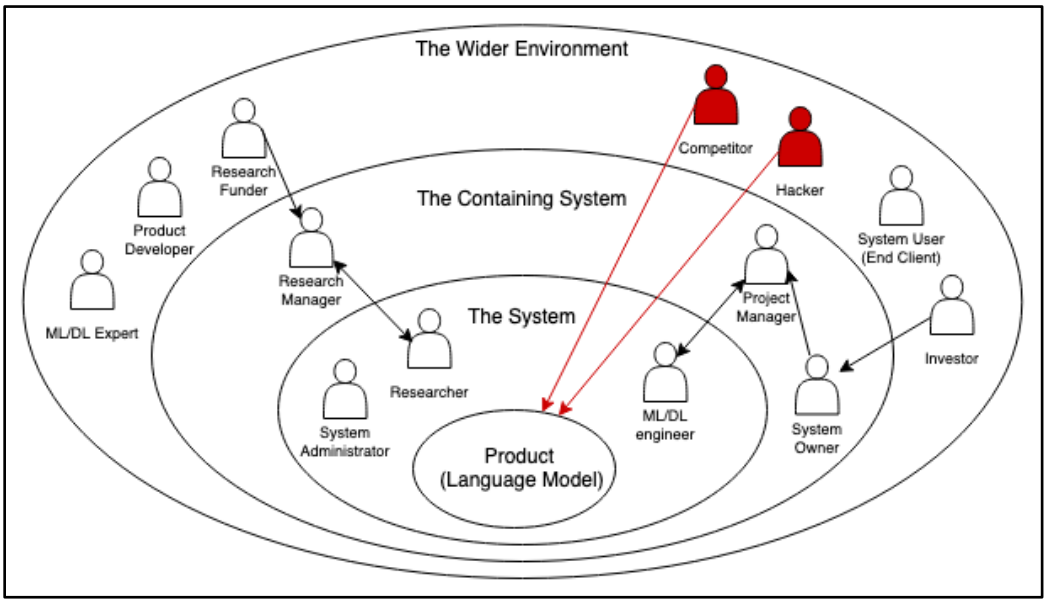


Figure 8 - Stakeholder Onion Model

4.3.2 Stakeholder Viewpoints

Figure 9 - Stakeholder Onion Model

Stakeholder	Role	Description
Researcher, ML/DL Engineer	Operator	Researching and building on the generative model.
System Administrator	Operator	Approving and deploying changes to the

		generative model
System Owner	Product Champion	Owns the project and oversees it.
Research Manager, Product Manager	Functional Beneficiary	Responsible for leading the research and engineering teams, and handling the requirement gathering, and task division to successfully build the project.
System User	User	Using the application to obtain generated text
Research Funder, Investor	Financial Beneficiary	Provides the monetary resources needed for the research and product
Product Developer	Developer	Responsible for developing the system (Front end and back end)
ML/DL Expert	Consultant	Domain and technological expert for providing additional support to improve the product
Competitor	Negative stakeholder	Other text data generating products that threaten the product as competitors
Hacker	Negative stakeholder	A threat, that can modify the system and training data, to generate useless data

Table 8 - Stakeholder Viewpoints

4.4 Selection of Requirement Elicitation Techniques/Methods

There are many methods used for gathering requirements and analysing them. This section focuses on the selection of certain methods in gathering requirements, and the reason for their selection.

4.4.1 Questionnaires

There are many systems that use language models and text generation, and there are plenty of users who have used such systems. The questionnaire mainly targets the end-users of such systems, to identify the issues that a user might face. The questionnaire also focuses on gathering requirements that are needed to research and build a text generation system: libraries, frameworks, languages, and techniques that would be required to successfully conduct research. The questionnaire was distributed to those who could have used systems that have text generation, such as students following computer science-related degrees, industry professionals, professors, etc.

4.4.2 Interviews

The crucial stakeholders in this research would be the domain experts, such as NLP engineers, data scientists, and researchers. Such stakeholders possess strong expertise in the research domain, with lots of industrial/academic experience, and can provide valuable insights in designing, building, and researching on a text generation system. Since such domain experts are not large enough to provide questionnaires, interviews are chosen to ask questions regarding the research and obtain insights from them. Thematic analysis was chosen to display the findings from the interviews.

The following were the themes chosen to obtain insights from the interviews: nature of data needed for generation, type of models or techniques that are used for generating text, and the basic functionalities of a working prototype to showcase the research.

4.4.3 Literature review and requirements reuse from existing research

Literature review provide plenty of information, insights, and limitations from the literature surrounding the research, and this is highly useful, as the literature is presented in many viewpoints of researchers. Also, various requirements that are needed to build the research/system can be reused from the existing works, and this can help speed up the research, saving a lot of time and cost, rather than building it from scratch.

4.4.4 Prototyping

Building an initial prototype can assist in gaining insights into where the research should focus on. Prototyping also speeds up the development, as there would be an initial working system to

improve upon. Since the research focuses on natural language processing, prototypes would be evaluated based on the metrics used.

4.4.5 Self-evaluation

Self-evaluation and self-brainstorming can assist in identifying requirements that might have eluded in the previous elicitation methods. This method can also assist in identifying new ways of improving the research and visualising the final end-product that can be obtained from the research.

4.5 Discussion of Results

This section focuses on the results obtained from the requirement elicitation methods and presents a discussion based on the results.

4.5.1 Findings from questionnaires

Question	Have you used products that use language models? (e.g., autocomplete, paraphrasing software, language translation, etc.)
Aim	To find out the percentage of people who have used systems that have language models.

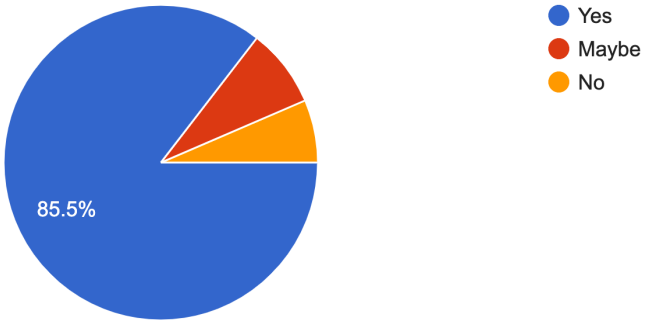
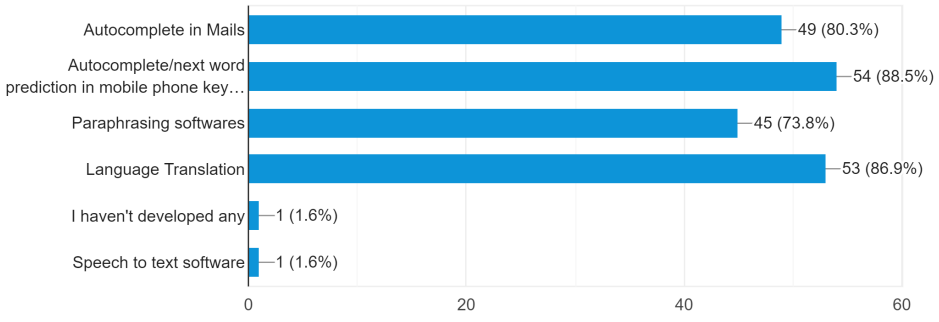
Observations	 Around 85% of the respondents have used systems that are powered by language models.
Conclusion	This question's aim is to filter out those who have not used a real-world system that has language models and text generation. The respondents, who have answered "No" will not be provided with the upcoming questions to answer.

Table 9 – Questionnaire Result 1

Question	What are the types of products you have come across?
Aim	To find out which systems, that are powered by language models, are used by the respondents.
Observations	 Majority of the respondents have used autocomplete software in their mobile keyboards, and for translation. And many have used autocomplete in mails and in paraphrasing software.

Conclusion	From this question, autocomplete software are one of the most used systems among the respondents.

Table 10 - Questionnaire Result 2

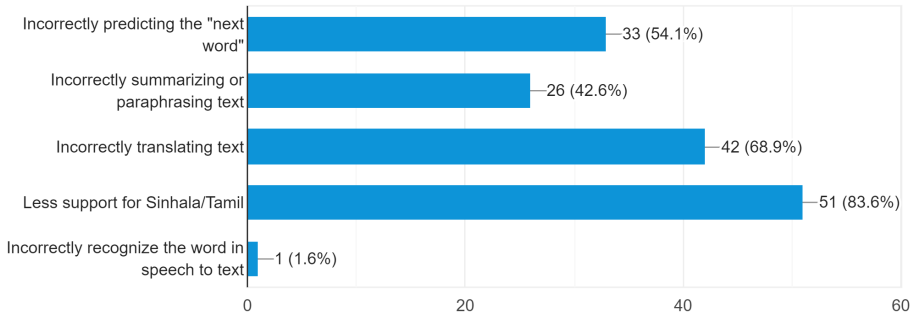
Question	What are the issues you have faced when using such products?
Aim	To find out the issues that the respondents face when using systems with language models or text generation.
Observations	 Less support for low-resource languages (such as Tamil and Sinhala) is a major issue that many respondents have faced, with incorrect translation and incorrect prediction of the next word are other issues that are present in such systems.
Conclusion	This question shows the issues that are faced by the people who use systems with language models or text generation, where less support for low-resource languages is a major issue, and that is followed by incorrect translation and incorrectly predicting the next word.

Table 11 - Questionnaire Result 3

Question	Which aspect should be improved when using such products?
Aim	To find out which aspect of the system with language models and text generation should be improved.

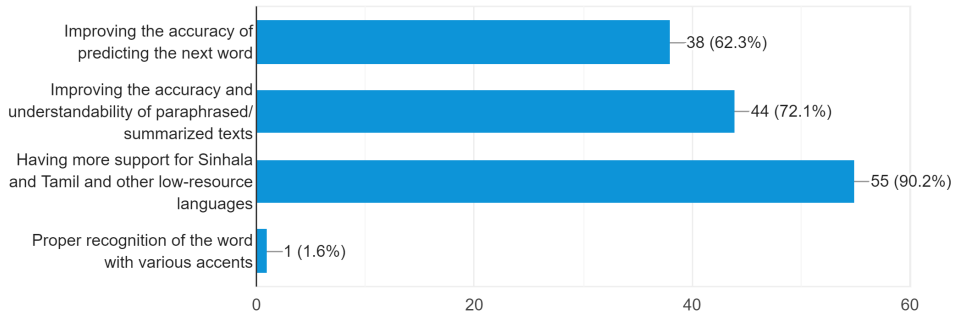
Observations	 Majority of the respondents state that more support and research should be provided for systems handling low-resource languages. Also, improving the accuracy of summarized texts, and improving the accuracy of next word predictions are stated by the respondents.
Conclusion	This question focuses on the major parts where research should focus on when handling systems with language models and text generation. Providing more support for low-resource languages is mostly favoured by the respondents, and the other areas where research should focus on are improving the accuracy of the next word prediction and improving text summarization.

Table 12 - Questionnaire Result 4

4.5.2 Findings from interviews

People with experience working on deep learning, machine learning, and natural language processing were selected for interviews. Two lecturers, three industrial experts, and three statistics special graduates were chosen for interviews, and the themes discussed in the previous section was used for interviewing them. The following are the findings obtained from the interviews:

Topic Discussed	Findings
Nature of data used	Using benchmark datasets for training and evaluating the model is always preferred when coming up with novel solutions. Organisations prefer training models on data that is related to the problem the organisation is trying to solve.

Need for generating data	NLP tasks require lots of data, and many domain-specific tasks face a lack of data. Therefore, generating more data from a set of small data would be highly beneficial.
Type of generation needed	Out of conditional and unconditional generations, conditional generation is preferred, as generation of text can be controlled, and desired outputs can be obtained. However, unconditional generation can also be useful when data is not labelled.
Techniques to follow when creating a text generation model	RNN-based models are easy to implement and understand, but currently, Transformers are becoming state-of-the-art. Also, using GANs for generating text can be explored.
Focus on low-resource language	In academics, it is better to focus on benchmark datasets, as it can provide much worth to the knowledge bank. However, low-resource languages should be focused on as well, especially when considering the industrial viewpoint.

Table 13 - Interview Findings

4.5.3. Findings from literature review and requirements reuse from existing research

Citation	Brief Description	Improvements	Limitations
(Guo et al., 2017)	Uses LSTM components in generating text, and a leaking feature to provide more information to the generator	State-of-the-art results in 2 datasets; works well in generating long sequences	Uses LSTM-components; takes a long time to train.
(Wu, Li and Yu, 2021)	Uses transformer models as components for generator and discriminator	Stable training, constant rewards were provided, performed well for	Large pre-trained models, relatively new research.

		conditional generation	
(Lin et al., 2018)	Uses LSTMs and a rank-based method for generating text.	No need of training a discriminator component	Not suitable for generating long sequences.
(Fedus, Goodfellow and Dai, 2018)	Uses LSTM to build a Seq2Seq generator and is training using RL	Mode collapse is reduced, reduces the risk of discriminator being trained too well	Uses LSTM-based components.
(Yu et al., 2017)	Uses RL and Monte-Carlo search to train the model	Started training GANS with RL for generating text	Generated texts with less diversiry, uses RNN-based components

Table 14 - LR Findings

4.5.4 Findings from prototyping

An initial prototype built using plain LSTM was initially reviewed. Even though it was quite easy to build such a model, the results were very poor. The generated sentence made little sense.

Then, a more advanced model was trained for evaluation. This model is an implementation of the LeakGAN paper, and the code is available publicly which can be used for research as it is under MIT license. The implementation and the results are present in the Implementation chapter. Training the LeakGAN model provided insights on generating meaningful texts using a GAN and enabled us to make architectural changes to such GAN-based text generation models.

4.5.5 Findings from self-evaluation

Following are some useful findings and ideas obtained from self-evaluation: a UI-based prototype of a text generation model would be highly useful for any end-users to understand and use the system effectively. The wireframes for such a system were designed (the diagram is present in the Design chapter). Also, brainstorming and self-evaluation helped a lot in converging on such a research idea, that is generating text using GANs.

4.6 Summary of Findings

Findings	**Q**	**I**	**LR**	**P**	**SE**
Need to improve the prediction of next word when building systems powered by Language Models.	x	x	x		
Low-resource languages, such as Sinhala and Tamil, have relatively less support when it comes to systems with Language Models.	x				
When building a prototype regarding text generation, it is preferred to use a benchmark dataset, to evaluate the model.		x	x		
A simple GUI is needed to simply explain to a user regarding the research.		x		x	x
Conditional generation is preferred, as the output can be controlled; and unconditional generation also has research opportunities when tackling the generation problem as a beginner.		x			

RNN-based text generation systems are currently present in many industries and implementing such a system is relatively easy.		x	x	x	
Transformers have become state-of-the-art in building language models and generating text.		x	x		
GANs has become a very good research domain for generating text.			x		

Table 15 - Summary of Findings

4.7 Context Diagram

The context diagram provides a high-level functionality of the system, depicting the interactions between the system and the user(s). Following is the context diagram for the research project:

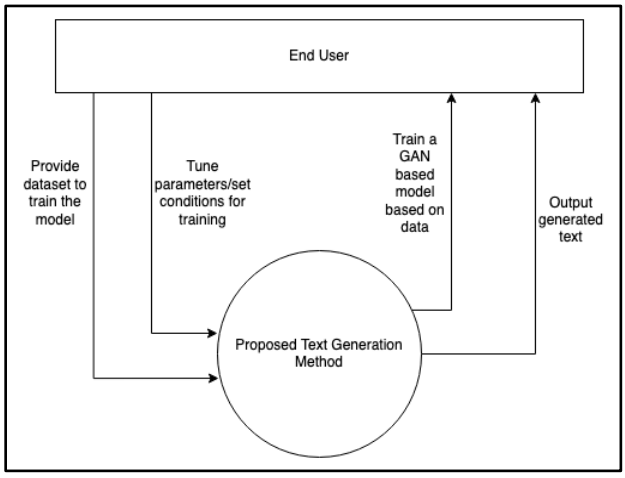


Figure 10 - Context Diagram

Figure 11 - Context Diagram

4.8 Use Case Diagram

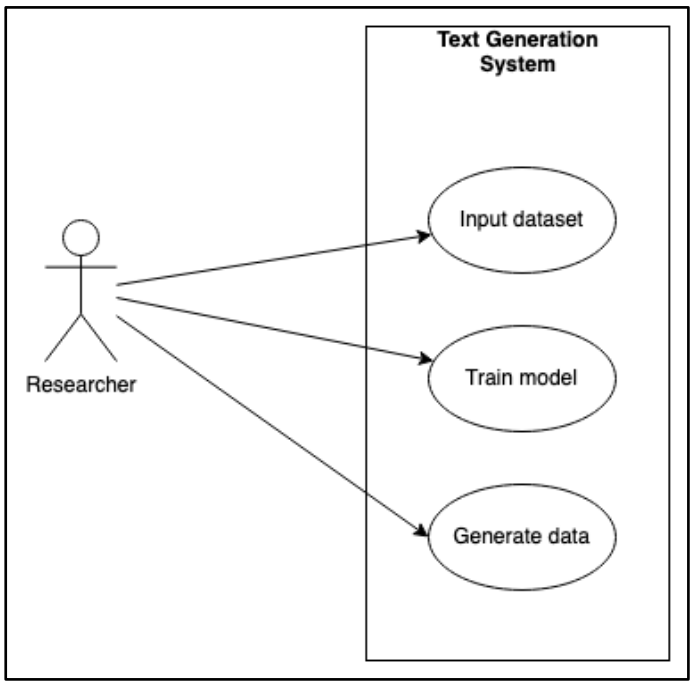


Figure 12 - Use Case Diagram

Figure 13 - Use Case Diagram

4.9 Use Case Description using Alistair Cockburn's template (only for the core use case/s)

Use Case	Input dataset
Description	Allows the user to upload a dataset to the system
Actor	End-user
Pre-conditions	None
Trigger	The end-user selects upload dataset option in the system
Flow of events	1. The user clicks on upload dataset option in the system 2. The user then uploads the text dataset to the system

	3. The system confirms that the dataset is uploaded.
Alternative flow of events	None
Exceptional flow of events	If the format of the dataset is not like the intended one, the user is shown the error message.
Success End Condition	The system displays that the dataset is successfully uploaded.
Failure End Condition	The dataset is not uploaded, and the system displays the relevant error message.

Table 16 - Use case for input dataset

Use Case	Train model
Description	Trains the model based on the dataset that is input
Actor	End-user
Pre-conditions	The dataset must be uploaded
Trigger	When the dataset is uploaded, and the user clicks on train the model.
Flow of events	1. The user clicks on train model option 2. The user can specify the training configurations 3. The system then trains the model 4. After training, the system confirms that the model has been trained.
Alternative flow of events	None
Exceptional flow of events	None
Success End Condition	The model has been successfully trained and the system displays

	the success message.
Failure End Condition	None

Table 17 - Use case for train model

Use Case	Generate data
Description	Generates text data after the model has been trained
Actor	End-user
Pre-conditions	The model must be trained
Trigger	After the model training, when the user clicks on generate data.
Flow of events	1. The user clicks on generate data option 2. The trained model generates data 3. The user can view and download the trained data
Alternative flow of events	None
Exceptional flow of events	None
Success End Condition	The user being able to view and download the generated dataset.
Failure End Condition	None

Table 18 - Use case for generate data

4.10 Functional Requirements (with prioritization)

The MoSCoW principle is used to determine the requirements, and the MoSCoW principle for the following research is as follows:

Must Have	Crucial requirements that need to be fulfilled in the research project.

Should Have	Important requirements that need to be fulfilled in the research project.
Could Have	Luxury requirements that could be implemented if needed in the research project.
Would Not Have	Requirements that are not required and will not be addressed in the research project.

Table 19 - MoSCoW principles

	Requirement	**Priority Level**
FR1	The end-user **must** be able to upload text data to the system.	Must Have
FR2	The end-user **must** be able to train the generative model by using the system.	Must Have
FR3	The end-user **must** be able to export/download the generated data.	Must Have
FR4	The end-user **should** be able to upload text data of low-resource language (Tamil or Sinhala) to the system.	Should Have
FR5	The end-user **could** be able to generate text by inputting specific keywords (conditional).	Could Have

Table 20 - Functional Requirements

4.11 Non-Functional Requirements

	Requirement	**Priority Level**
NFR1	The functionalities of the model **must** be easily visualized in the GUI.	Must Have

NFR2	The GUI **must** be able to run on a web browser (e.g., Chrome, Firefox, etc.).	Must Have
NFR3	The research **must** be easy to reproduce and extend on.	Must Have
NFR4	The generated text **should** be comprehensible.	Should Have
NFR5	A clear documentation **could** be provided.	Could Have

Table 21 - Non-functional requirements

4.12 Chapter Summary

In the Software Requirement Specification chapter, the requirement elicitation methodologies, the stakeholders of the system, the requirements of the system were discussed and evaluated. Rich picture diagram visualised how the system would interact with the big environment, and the stakeholder onion model pictures the personalities who influence the system. Questionnaires, interviews, literature reviews, prototyping and self-evaluation were chosen as requirements elicitation methods. The context diagram and use-case diagram depict how the end-user would interact with the system. The chapter concludes with the functional and non-functional requirements systems, determined by the MoSCow principles.

CHAPTER 5: SOCIAL, LEGAL, ETHICAL AND PROFESSIONAL ISSUES

5.1 Chapter Overview

The chapter is based on the social, legal, ethical, and professional issues that might have risen throughout the research, and how steps were taken to mitigate them.

5.2 SLEP Issues and Mitigation

5.2.1 Social

- The data collected from questionnaires at certain points in the research were completely anonymous: no user data was saved and only the data needed for the research was obtained.
- The responses of the questionnaires were not displayed in the thesis; only the analysis results obtained from the responses were displayed.

5.2.2 Legal

- The tools and technologies used for the research, Python, PyTorch, TextGAN-PyTorch, etc., are all licensed under the GPL and MIT license.
- The personal data from any survey respondents were not recorded, therefore keeping their identities private.
- The data used for the training is open-sourced and publicly available data. No kind of scraping nor illegal procuring of data was used during data collection.
- Interviews were done through video, and answers were recorded in a Google Form. The identities of the interviewees are not published, unless specifically requested by the interviewee.

5.2.3Ethical

- The questionnaires provided to the respondents were made aware about the research prior to them answering.
- No plagiarism nor falsification of any component of the thesis was made.

5.2.4 Professional

- The tools and software used to develop the research were legal and non-pirated. Open-sourced licensed and student licensed software were also used.
- No data was fabricated nor falsified to conduct the research.
- The whole prototype was trained and developed in a secure cloud environment, with updated security patches.
- The limitations of the project were observed and documented and conveyed to the evaluators.

5.3 Chapter Summary

The main issues in the social, legal, ethical, and professional viewpoints discussed and the way they were tackled during the research was explained.

CHAPTER 6: SYSTEM ARCHITECTURE & DESIGN

6.1 Chapter Overview

This chapter reflects on the initial design of the research project. Initially, the design goals of the system would be explained, which would be followed throughout the development. Then, a system architecture, that is suitable for the research project, is proposed, along with a high-level design diagram. Then, the low-level design diagram is proposed using suitable diagrams. The following designs and diagrams are subject to change in the final thesis submission.

6.2 Design Goals

Design Goal	Description
Correctness	The system must be able to generate text that looks human-written, and not meaningless sentences.
Performance	The model training must happen without any errors, and the resources must be used efficiently when training.
Reusability	The research must be designed in a way where another researcher can extend upon this research. Also, the system must be able to assist in solving other major tasks in NLP by generating useful textual data.
Configurability	The system must be easily configurable, so a researcher or domain expert can extend upon the system, and a developer can

	solve issues that might arise.

Table 22 - Design Goals

6.3 System Architecture Design

Considering the design goals and the requirements explained in the earlier section, a **Tiered** architecture is chosen to represent the system's high-level design. The tiers, frontend, backend, and data were designed based on real-world scenarios, where it would be easy to deploy each tier separately and have connections between them.

6.3.1 Layered Architecture/Tiered Architecture/etc.

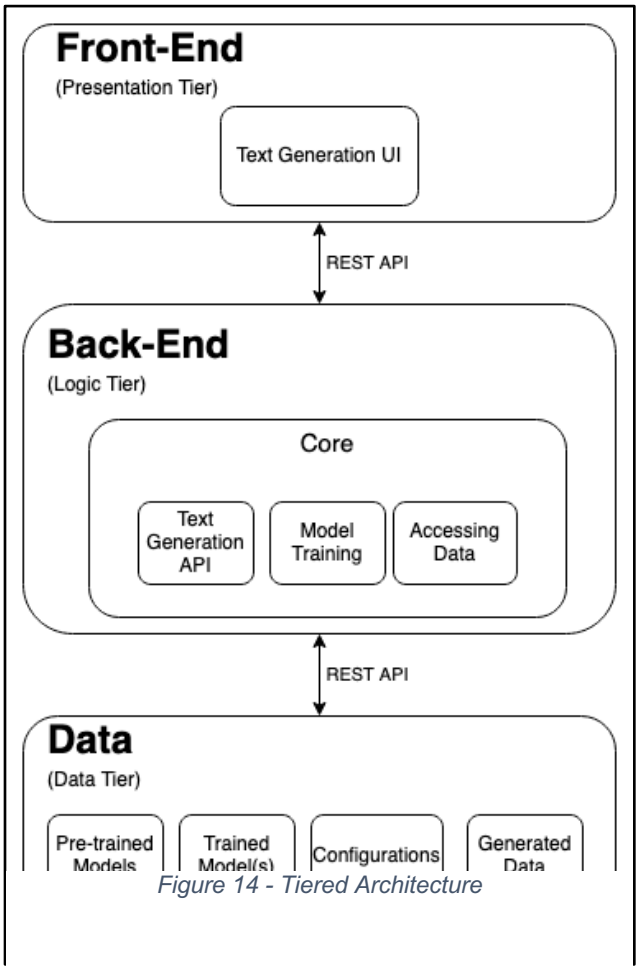


Figure 14 - Tiered Architecture

Figure 15 - Tiered Architecture

- Front-end: A graphical user interface that allows the user to interact with the system. The front-end would be a web application, that runs on a web browser. This can be deployed on a low-cost server, without a need for GPU.
- Back-end: The server contains the core text generation API, where the server takes requests from the front-end. To deploy this tier, a dedicated, high-end GPU server is used, to enable faster model training.
- Data: a persistent server, to store the pre-trained and trained models, and the database, that contains training data, generated data, etc.

6.4 System Design

6.4.1 Choice of Design Paradigm

Since the research project is a data science project, **SSADM** (Structured Systems Analysis and Design Method) is used as the design paradigm. SSADM allows to break up the development into smaller modules and steps, so each module can focus on improving the output of the research. SSADM also is good for managing potential risks that can be faced during development, making the project manageable with maintainable codes and components, and more likely to meet the requirements that are set.

The code for the model is built using object-oriented programming, as classes are used to code the generator, discriminator, and other modules.

6.4.2 Level-0 and Level-1 Data Flow Diagram

The level-0 data flow diagram describes the high-level overview of how the system is used by the end user, where the flow of data is depicted. The level-1 data flow diagram expands on the “Generate Text” Process and takes a deeper look on how the text generation occurs.

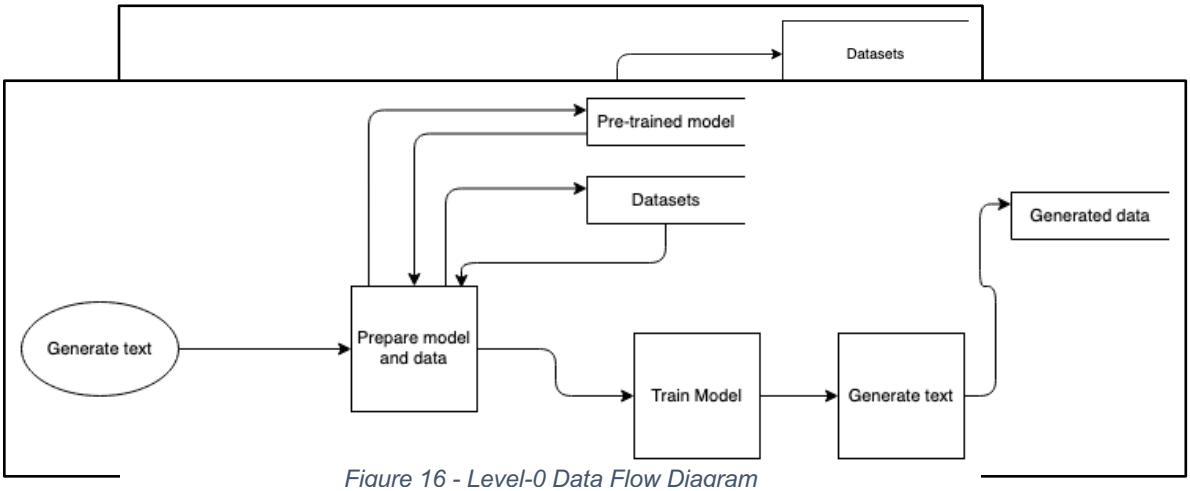


Figure 16 - Level-0 Data Flow Diagram
Figure 18 - Level-1 Data Flow Diagram

Figure 19 - Level-1 Data Flow Diagram

6.4.3 Algorithmic Flow Diagram

The following diagram gives an overview of how the algorithm works in generating textual data in a high-level view. The flow starts at inputting dataset, and the dataset would be pre-processed, and then training would be performed and finally, the generated text data would be obtained.

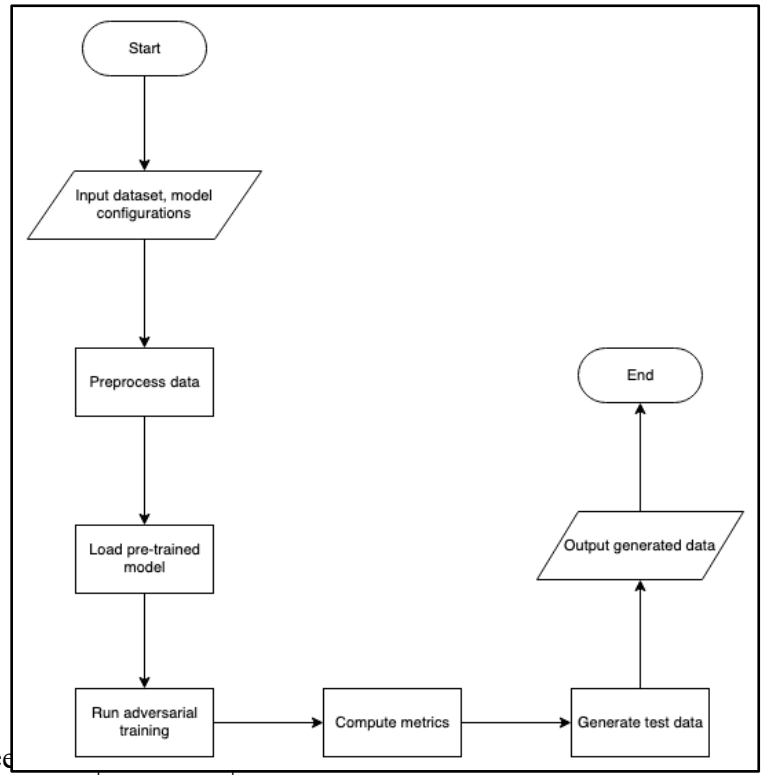


Figure 20 - Algorithmic Flow

6.4.4 Model Diagram

Following is the architecture of the model that is used for research. The implementation is from the LeakGAN paper. The specialty of this model is that, before the discriminator arrives at a verdict regarding the generated text, the features used for the classification is leaked to the generator to improve the quality of the generated text.

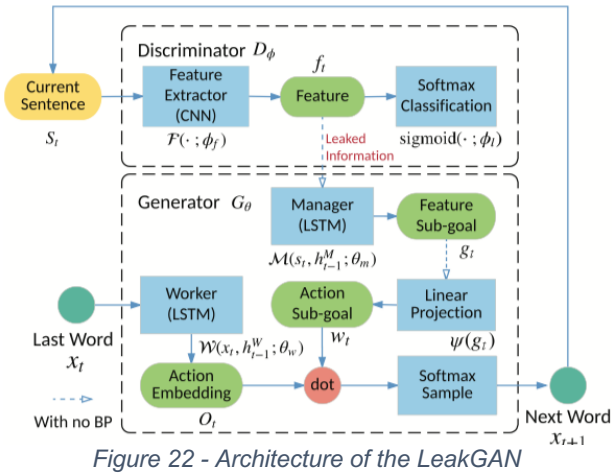


Figure 22 - Architecture of the LeakGAN

Figure 23 - Architecture of the LeakGAN

6.4.5 UI Design – Use low fidelity wireframes/high fidelity prototype

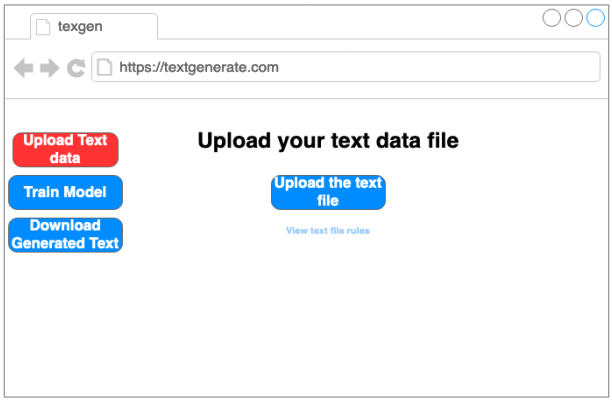


Figure 24 - UI Wireframe 1

Figure 25 - UI Wireframe 1

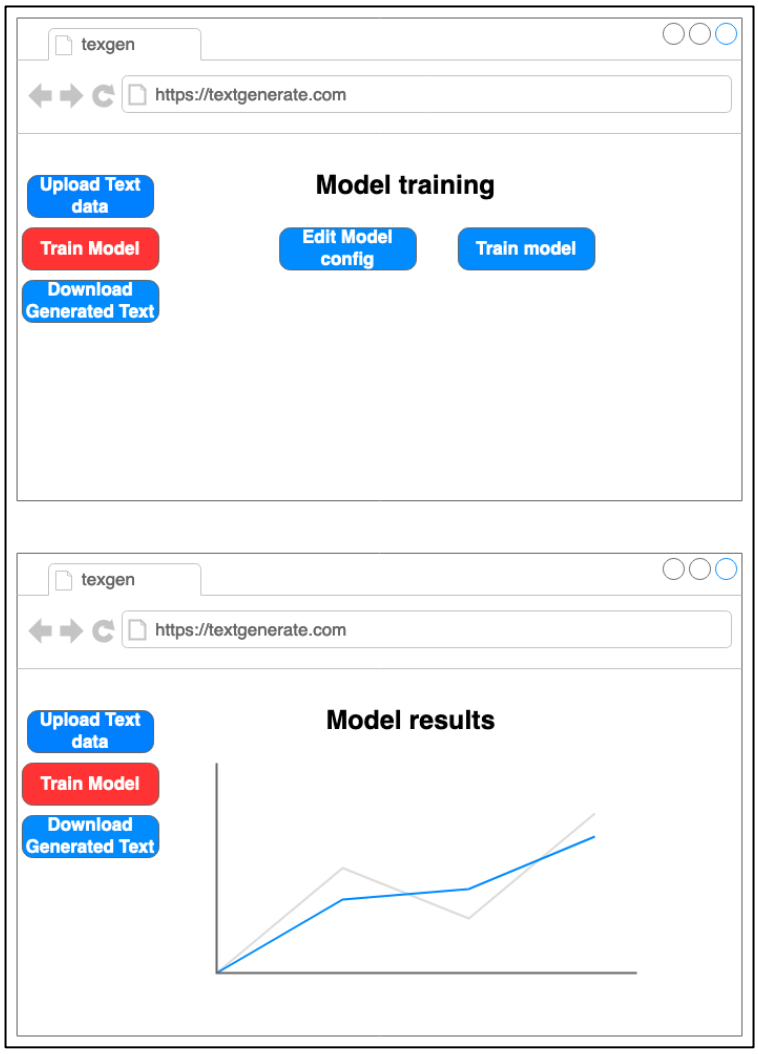


Figure 26 - UI Wireframe 2

Figure 27 - UI Wireframe 2

6.5 Chapter Summary

The chapter focused on the design aspects of the research project. Initially, the design goals of the system were explained. Tier architecture was chosen to display the system's high-level functionalities. SSADM was chosen as the design paradigm, and the relevant diagrams regarding the SSADM and the model were presented, along with the proposed wireframes for the system.

CHAPTER 7: IMPLEMENTATION (THINK OF ANYTHING ELSE WHICH IS RELEVANT AND ADD IT)

7.1 Chapter Overview

The final chapter of the document focuses on the implementation of the initial prototype. The technology stack, that would be used to develop the prototype is discussed, followed by important code snippets of the core functionalities, and would conclude with the author's self-evaluation of the initial prototype.

7.2 Technology Selection

7.2.1 Technology Stack

As proposed in the high-level architecture design, the implementation would consist of three tiers: front-end, back-end, and data; and each tier would need specific technologies when implementing them. The following diagram provides an overview of the technologies used in the tiers.

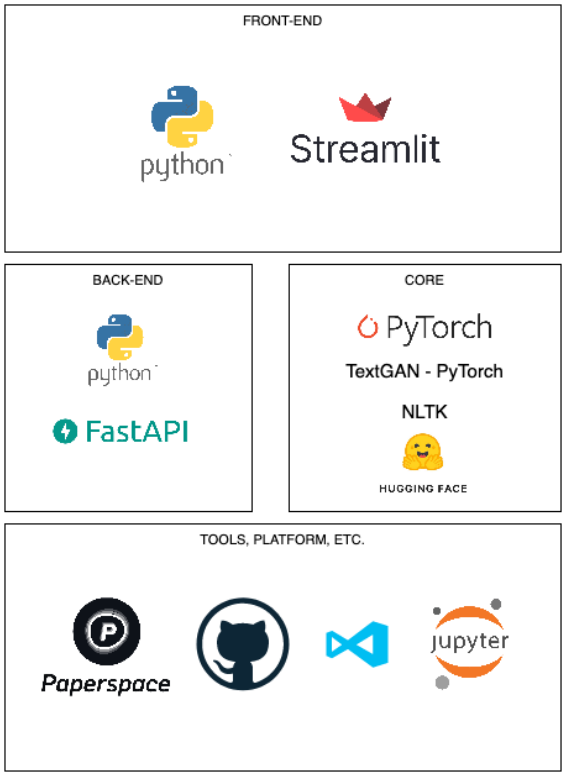


Figure 28 - Technology Stack

Figure 29 - Technology Stack

7.2.2 Data Selection (If Data Science project)

One of the main focuses of the research is unconditional text generation, therefore ImageCOCO captions (Chen et al., 2015) and EMNLP News datasets(statmt.org, 2017) were considered to train the model. ImageCOCO is a famous dataset for training image-based models, and ImageCOCO captions provide captions for the images, so only the captions can be taken for natural language processing tasks. ImageCOCO captions have been used in research regarding short text generation, and have been a benchmark dataset, therefore ImageCOCO captions have been selected as one of the datasets for the following research.

EMNLP (Empirical Methods in Natural Language Processing) is a conference held annually regarding research on NLP and EMNLP News dataset has been one of the benchmark datasets in training medium text generation models; therefore, EMNLP News dataset too was chosen as one of the datasets for this research.

7.2.3 Selection of Development Framework

Framework	Reason for choosing
PyTorch	PyTorch is a flexible deep learning framework, that can be used for building models for research. A main reason for selecting PyTorch is that many of the modern research ad publishments have implementations in PyTorch, which can be re-used. Also, PyTorch is more pythonic compared to its direct competitor, TensorFlow.
TextGAN-PyTorch	A GAN-based framework for building text generation models and built using PyTorch. The reason for selecting this framework is that it contains the base models that have been published, with benchmarking and data, therefore, it is relatively easy to compare the results when making changes.
FastAPI	A lightweight, fast framework for building APIs, FastAPI is chosen to build the back-end server. It has fewer bugs, when compared to other frameworks like Flask and Django, and easy to implement.

Streamlit	An open-sourced framework for building apps faster and easier, Streamlit is chosen to build the front-end. Streamlit is used by many data scientists and machine learning engineers to quickly develop a web-app for their projects.

Table 23 - Development Framework

7.2.4 Programming Language

Out of the languages available for building data science projects (Python, R, Java, and C++), Python is chosen as the programming language for this research. Python is one of the most supported languages currently, with plenty of community and open-sourced frameworks and libraries present to work on various tasks. All the afore mentioned frameworks and libraries in the previous section are built for Python.

7.2.5 IDE

IDE	Reason for choosing
VS Code	A lightweight code editor that can support almost any types of files, and with plenty of add-ons created by the community. VS Code is used in the project to search, view, and edit different types of files.
PyCharm	A dedicated Python IDE, that has plenty of support for Python-based coding, from building servers to data science projects. PyCharm is chosen as it provides many supports, such as documentation, indexing of files, smart completion when handling Python-based code base.
Jupyter Lab/Notebook	An open-sourced environment to quickly research, build, visualize, and test ML or DL models in notebooks.

Table 24 -IDEs

7.2.6 Summary of Technology Selection (Table format)

Component	Selection
Programming Language	Python

Front-end Library	Streamlit
Back-end Library	FastAPI
Deep Learning Framework	PyTorch
GAN Framework	TextGAN-PyTorch
Data	ImageCOCO Captions, EMNLP News
IDEs	VS Code, PyCharm, Jupyter Notebook

Table 25 - Summary of Technology Selection

7.3 Implementation of Core Functionalities

The core functionalities of the research project are implemented using the TextGAN-PyTorch library (WilliamLam, 2022), which provides re-usable GAN components for generating text. The following research attempts to implement a modified version of the LeakGAN (Guo et al., 2017), which is a state-of-the-art implementation in generating text, and has good performance in the ImageCOCO Captions dataset. For this research, modifications were made to the architecture of the LeakGAN model, and the results were monitored after training them.

This section would consist of the important code needed for running the LeakGAN model: the generator, the discriminator, the instructor file, the run file, and the data pre-processing file.

7.3.1 LeakGAN Generator

The following snippets represent the code for the generator of the LeakGAN. The first snippet indicates the variables used to build the LeakGAN. The second snippet represents the forward pass of the generator. The “worker” and the “manager” module would be replaced by different models to conduct the research.

```
class LeakGAN_G(nn.Module):
    def __init__(self, embedding_dim, hidden_dim, vocab_size, max_seq_len, padding_idx, goal_size,
                 step_size, gpu=False):
        super(LeakGAN_G, self).__init__()
        self.name = 'leakgan'

        self.hidden_dim = hidden_dim
        self.embedding_dim = embedding_dim
        self.max_seq_len = max_seq_len
        self.vocab_size = vocab_size
        self.padding_idx = padding_idx
        self.goal_size = goal_size
        self.goal_out_size = goal_out_size  # equals to total_num_filters
        self.step_size = step_size
        self.gpu = gpu
        self.temperature = 1.5

        self.embeddings = nn.Embedding(vocab_size, embedding_dim, padding_idx=padding_idx)
        self.worker = nn.LSTM(embedding_dim, hidden_dim)
        self.manager = nn.LSTM(goal_out_size, hidden_dim)

        self.work2goal = nn.Linear(hidden_dim, vocab_size * goal_size)
        self.mana2goal = nn.Linear(hidden_dim, goal_out_size)
        self.goal2goal = nn.Linear(goal_out_size, goal_size, bias=False)

        self.goal_init = nn.Parameter(torch.rand((cfg.batch_size, goal_out_size)))

        self.init_params()
```

Figure 30 - LeakGAN Generator

*Figure 31 - LeakGAN Generator forward pass**Figure 32 - LeakGAN Generator*

```
class LeakGAN_G(nn.Module):
    def __init__(self, embedding_dim, hidden_dim, vocab_size, max_seq_len, padding_idx, goal_size,…

    def forward(self, idx, inp, work_hidden, mana_hidden, feature, real_goal, no_log=False, train=False):
        """
        Embeds input and sample on token at a time (seq_len = 1)

        :param idx: index of current token in sentence
        :param inp: [batch_size]
        :param work_hidden: 1 * batch_size * hidden_dim
        :param mana_hidden: 1 * batch_size * hidden_dim
        :param feature: 1 * batch_size * total_num_filters, feature of current sentence
        :param real_goal: batch_size * goal_out_size, real_goal in LeakGAN source code
        :param no_log: no log operation
        :param train: if train

        :return: out, cur_goal, work_hidden, mana_hidden
            - out: batch_size * vocab_size
            - cur_goal: batch_size * 1 * goal_out_size
        """
        emb = self.embeddings(inp).unsqueeze(0)  # 1 * batch_size * embed_dim

        # Manager
        mana_out, mana_hidden = self.manager(feature, mana_hidden)  # mana_out: 1 * batch_size * hidden_dim
        mana_out = self.mana2goal(mana_out.permute([1, 0, 2]))  # batch_size * 1 * goal_out_size
        cur_goal = F.normalize(mana_out, dim=-1)
        _real_goal = self.goal2goal(real_goal)  # batch_size * goal_size
        _real_goal = F.normalize(_real_goal, p=2, dim=-1).unsqueeze(-1)  # batch_size * goal_size * 1

        # Worker
        work_out, work_hidden = self.worker(emb, work_hidden)  # work_out: 1 * batch_size * hidden_dim
        work_out = self.work2goal(work_out).view(-1, self.vocab_size,
                                                 self.goal_size)  # batch_size * vocab_size * goal_size

        # Sample token
        out = torch.matmul(work_out, _real_goal).squeeze(-1)  # batch_size * vocab_size

        # Temperature control
        if idx > 1:
            if train:
                temperature = 1.0
            else:
                temperature = self.temperature
        else:
            temperature = self.temperature

        out = temperature * out

        if no_log:
            out = F.softmax(out, dim=-1)
        else:
            out = F.log_softmax(out, dim=-1)

        return out, cur_goal, work_hidden, mana_hidden
```

Figure 33 - LeakGAN Generator forward pass

Figure 34 - LeakGAN discriminatorFigure 35 - LeakGAN Generator forward pass

7.3.2 LeakGAN Discriminator

The following is the code for the discriminator, which detects if the generated sequence is fake or real-like. A CNN model is used to extract the features of the generated text, that is then used for the classification.

```
class CNNDiscriminator(nn.Module):
    def __init__(self, embed_dim, vocab_size, filter_sizes, num_filters, padding_idx, gpu=False,
                 dropout=0.2):
        super(CNNDiscriminator, self).__init__()
        self.embedding_dim = embed_dim
        self.vocab_size = vocab_size
        self.padding_idx = padding_idx
        self.feature_dim = sum(num_filters)
        self.gpu = gpu

        self.embeddings = nn.Embedding(vocab_size, embed_dim, padding_idx=padding_idx)
        self.convs = nn.ModuleList([
            nn.Conv2d(1, n, (f, embed_dim)) for (n, f) in zip(num_filters, filter_sizes)
        ])
        self.highway = nn.Linear(self.feature_dim, self.feature_dim)
        self.feature2out = nn.Linear(self.feature_dim, 2)
        self.dropout = nn.Dropout(dropout)

        self.init_params()

    def forward(self, inp):
        """
        Get final predictions of discriminator
        :param inp: batch_size * seq_len
        :return: pred: batch_size * 2
        """
        feature = self.get_feature(inp)
        pred = self.feature2out(self.dropout(feature))

        return pred

    def get_feature(self, inp):
        """
        Get feature vector of given sentences
        :param inp: batch_size * max_seq_len
        :return: batch_size * feature_dim
        """
        emb = self.embeddings(inp).unsqueeze(1)  # batch_size * 1 * max_seq_len * embed_dim
        convs = [F.relu(conv(emb)).squeeze(3) for conv in self.convs]  # [batch_size * num_filter * length]
        pools = [F.max_pool1d(conv, conv.size(2)).squeeze(2) for conv in convs]  # [batch_size * num_filter]
        pred = torch.cat(pools, 1)  # tensor: batch_size * feature_dim
        highway = self.highway(pred)
        pred = torch.sigmoid(highway) * F.relu(highway) + (1. - torch.sigmoid(highway)) * pred  # highway

        return pred

    def init_params(self):
        for param in self.parameters():
            if param.requires_grad and len(param.shape) > 0:
                stddev = 1 / math.sqrt(param.shape[0])
                if cfg.dis_init == 'uniform':
                    torch.nn.init.uniform_(param, a=-0.05, b=0.05)
                elif cfg.dis_init == 'normal':
                    torch.nn.init.normal_(param, std=stddev)
```

Figure 36 - LeakGAN discriminator

Figure 37 - LeakGAN instructorFigure 38 - LeakGAN discriminator

7.3.3 LeakGAN Instructor

The instructor code snippet for the model, which controls the two training phases of the GAN model: a pre-training phase where the generator and discriminator are separately trained, and the adversarial training phase, where both generator and discriminator are trained together.

```
class LeakGANInstructor(BasicInstructor):
    def __init__(self, opt):
        super(LeakGANInstructor, self).__init__(opt)

        # generator, discriminator
        self.gen = LeakGAN_G(cfg.gen_embed_dim, cfg.gen_hidden_dim, cfg.vocab_size, cfg.max_seq_len,
                             cfg.padding_idx, cfg.goal_size, cfg.step_size, cfg.CUDA)
        self.dis = LeakGAN_D(cfg.dis_embed_dim, cfg.vocab_size, cfg.padding_idx, gpu=cfg.CUDA)
        self.init_model()

        # optimizer
        mana_params, work_params = self.gen.split_params()
        mana_opt = optim.Adam(mana_params, lr=cfg.gen_lr)
        work_opt = optim.Adam(work_params, lr=cfg.gen_lr)

        self.gen_opt = [mana_opt, work_opt]
        self.dis_opt = optim.Adam(self.dis.parameters(), lr=cfg.dis_lr)

    def _run(self):
        for inter_num in range(cfg.inter_epoch):
            self.log.info('>>> Interleaved Round %d...' % inter_num)
            self.sig.update()  # update signal
            if self.sig.pre_sig:
                # ===DISCRIMINATOR PRE-TRAINING===
                if not cfg.dis_pretrain:
                    self.log.info('Starting Discriminator Training...')
                    self.train_discriminator(cfg.d_step, cfg.d_epoch)
                    if cfg.if_save and not cfg.if_test:
                        torch.save(self.dis.state_dict(), cfg.pretrained_dis_path)
                        print('Save pre-trained discriminator: {}'.format(cfg.pretrained_dis_path))

                # ===GENERATOR MLE TRAINING===
                if not cfg.gen_pretrain:
                    self.log.info('Starting Generator MLE Training...')
                    self.pretrain_generator(cfg.MLE_train_epoch)
                    if cfg.if_save and not cfg.if_test:
                        torch.save(self.gen.state_dict(), cfg.pretrained_gen_path)
                        print('Save pre-trained generator: {}'.format(cfg.pretrained_gen_path))
            else:
                self.log.info('>>> Stop by pre_signal! Skip to adversarial training...')
                break

        # ===ADVERSARIAL TRAINING===
        self.log.info('Starting Adversarial Training...')
        self.log.info('Initial generator: %s' % (str(self.cal_metrics(fmt_str=True))))

        for adv_epoch in range(cfg.ADV_train_epoch):
            self.log.info('-----\nADV EPOCH %d\n-----' % adv_epoch)
            self.sig.update()
            if self.sig.adv_sig:
                self.adv_train_generator(cfg.ADV_g_step)  # Generator
                self.train_discriminator(cfg.ADV_d_step, cfg.ADV_d_epoch, 'ADV')  # Discriminator

                if adv_epoch % cfg.adv_log_step == 0 or adv_epoch == cfg.ADV_train_epoch - 1:
                    if cfg.if_save and not cfg.if_test:
                        self._save('ADV', adv_epoch)
            else:
```

Figure 39 - LeakGAN instructor

Figure 40 - LeakGAN run fileFigure 41 - LeakGAN instructor

7.3.4 LeakGAN run file

The code snippet that is used to run the LeakGAN file, which initialises the variables that are used for training the GAN model.

```
# ===Program===
if_test = int(False)
run_model = 'leakgan'
CUDA = int(True)
oracle_pretrain = int(True)
gen_pretrain = int(False)
dis_pretrain = int(False)
MLE_train_epoch = 8
ADV_train_epoch = 100 #200
inter_epoch = 10
tips = 'LeakGAN experiments'

# ===Oracle  or Real===
if_real_data = [int(False), int(True), int(True)]
dataset = ['oracle', 'image_coco', 'emnlp_news']
vocab_size = [5000, 0, 0]

# ===Basic Param===
data_shuffle = int(False)
model_type = 'vanilla'
gen_init = 'normal'
dis_init = 'uniform'
samples_num = 10000
batch_size = 64
max_seq_len = 20
gen_lr = 0.0015
dis_lr = 5e-5
pre_log_step = 1
adv_log_step = 1

# ===Generator===
ADV_g_step = 1
rollout_num = 4
gen_embed_dim = 32
gen_hidden_dim = 32
goal_size = 16
step_size = 4

# ===Discriminator===
d_step = 5
d_epoch = 3
ADV_d_step = 5
ADV_d_epoch = 3
dis_embed_dim = 64
dis_hidden_dim = 64

# ===Metrics===
use_nll_oracle = int(True)
use_nll_gen = int(True)
use_nll_div = int(True)
use_bleu = int(True)
```

Figure 42 - LeakGAN run file

Figure 43 - Text Pre-process codeFigure 44 - LeakGAN run file

7.3.5 Text pre-process code

Following is the code snippet to pre-process the dataset file and create tokens and word vectors that are fed into the model.

```
def get_tokenlized(file):
    """tokenlize the file"""
    tokenlized = list()
    with open(file) as raw:
        for text in raw:
            text = nltk.word_tokenize(text.lower())
            tokenlized.append(text)
    return tokenlized


def get_word_list(tokens):
    """get word set"""
    word_set = list()
    for sentence in tokens:
        for word in sentence:
            word_set.append(word)
    return list(set(word_set))


def get_dict(word_set):
    """get word2idx_dict and idx2word_dict"""
    word2idx_dict = dict()
    idx2word_dict = dict()

    index = 2
    word2idx_dict[cfg.padding_token] = str(cfg.padding_idx)  # padding token
    idx2word_dict[str(cfg.padding_idx)] = cfg.padding_token
    word2idx_dict[cfg.start_token] = str(cfg.start_letter)  # start token
    idx2word_dict[str(cfg.start_letter)] = cfg.start_token

    for word in word_set:
        word2idx_dict[word] = str(index)
        idx2word_dict[str(index)] = word
        index += 1
    return word2idx_dict, idx2word_dict


def text_process(train_text_loc, test_text_loc=None):
    """get sequence length and dict size"""
    train_tokens = get_tokenlized(train_text_loc)
    if test_text_loc is None:
        test_tokens = list()
    else:
        test_tokens = get_tokenlized(test_text_loc)
    word_set = get_word_list(train_tokens + test_tokens)
    word2idx_dict, idx2word_dict = get_dict(word_set)

    if test_text_loc is None:
        sequence_len = len(max(train_tokens, key=len))
    else:
        sequence_len = max(len(max(train_tokens, key=len)), len(max(test_tokens, key=len)))

    return sequence_len, len(word2idx_dict)
```

Figure 45 - Text Pre-process code

Figure 46 - Text Pre-process code

7.4 Implementation of UI

Streamlit Library was used to build a simple frontend to showcase the research. The following are the snapshots of the UI.

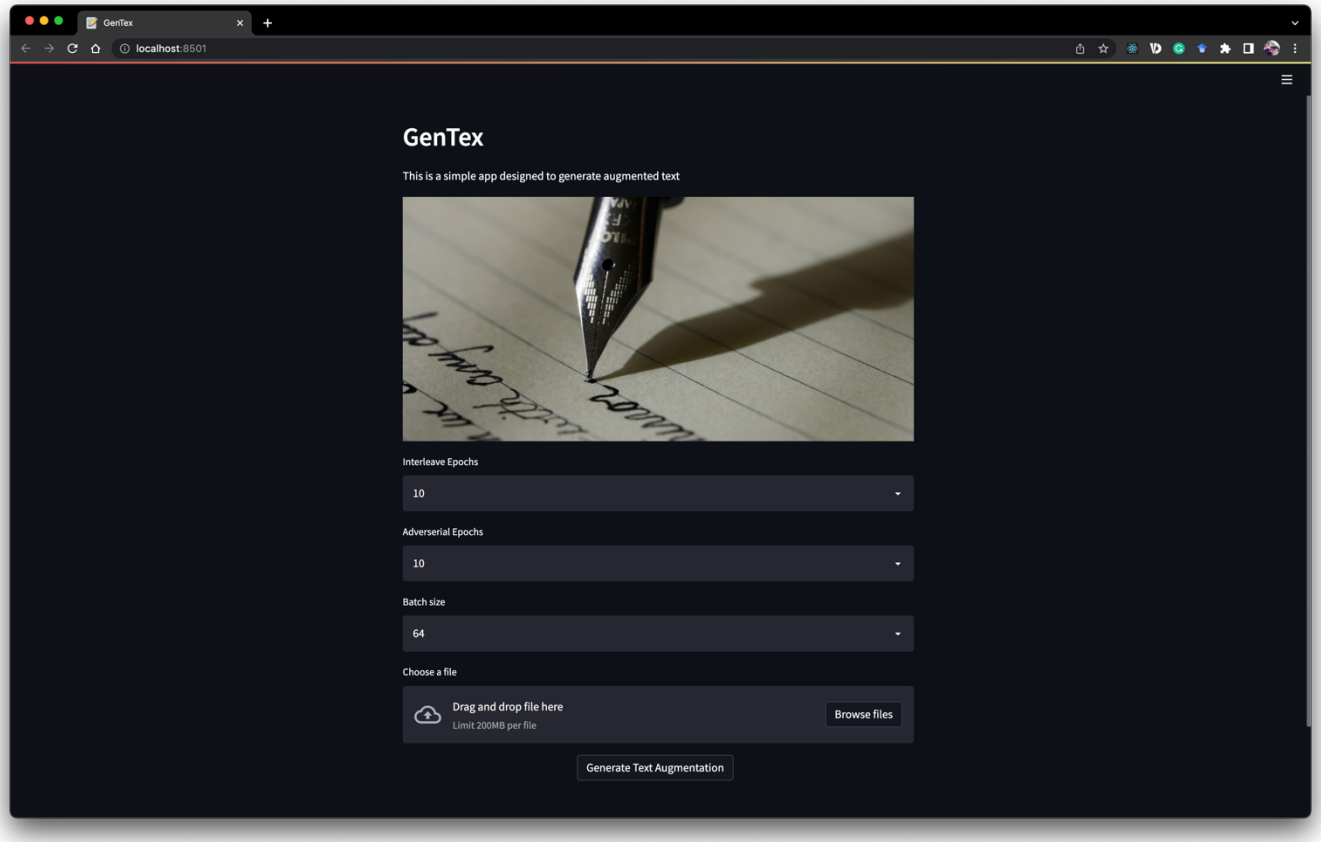


Figure 47 - UI of prototype

Figure 48 - UI of prototype

7.5 Chapter Summary

The steps on how the implementation was done was explained in this chapter. The reasons for choosing certain stacks to implement the research were explained, and the code snippets that were used to develop the core component of the research were provided.

CHAPTER 8: TESTING

8.1 Chapter Overview

This chapter focuses on testing the core functionality of the research. Model testing, benchmarking, and other testing methods would be discussed in detail in this chapter.

8.2 Objectives and Goals of Testing

The main objective of the testing process is to certify that the model and system are performing to the level set by the requirements that were gathered. Following are the goals for the testing process of the following research:

- To check whether the model is working properly, and the best outcomes are being achieved.
- To perform benchmarking tests on the model, and test how it is performing with other works.
- To check whether the system satisfies the functional requirements and non-functional requirements.

8.3 Testing Criteria

The system would be tested based on the following criteria:

1. Functional quality: To test how the system functions well, based on the initial design and the functional requirements.
2. Structural quality: To test the non-functional requirements of the system.

8.4 Model Testing

The datasets, ImageCOCO Captions and EMNLP 2017 News were chosen to train the model. However, due to budget constraints, only the ImageCOCO Captions were trained and evaluated, for short text generation. The datasets would assist in benchmarking as well, as the data that were chosen were used for other research.

The models are based on different architectures of the proposed variant of the LeakGAN model. The "worker" and "manager" components in the generator and the training of the discriminator were modified to obtain the optimal result from the training. The BLEU metric was chosen as the main metric to evaluate the quality of the generated sentences, and the Negative Log-Likelihood loss is chosen to monitor the stability of the GAN during training.

8.4.1 Testing Model on ImageCOCO Captions dataset

8.4.1.1 Base model with LSTM components for Worker and Manager components

The base model, specified by the LeakGAN paper, was trained with the ImageCOCO captions dataset. Both the worker and manager components of this model are LSTMs. The interleaved training epochs are 10 and the adversarial epochs are 10. The original epochs for interleaved training were reduced from 15 to 10 to accommodate the training process without extensive use of monetary resources. In each interleaved epoch, during the pre-training phase, the discriminator would face a 5 step training, and the generator would face an 8 step training. During the adversarial training, the discriminator would face a 5 step training, and the generator would face a 1 step training. The results can be found in the Table 26.

8.4.1.2 Model with GRU component for Manager and LSTM for Worker

A new variant of the LeakGAN was designed newly for the research. The Manager component was modified with a GRU, and the Worker was the same as the base model. The other parameters of the model were as same as the base one. The results can be found in the Table 26.

8.4.1.3 Model with LSTM component for Manager and GRU for Worker

A new variant of the LeakGAN was designed newly for the research. The Worker component was modified with a GRU, and the Manager was the same as the base model. The other parameters of the model were as same as the base one. The results can be found in the Table 26.

8.4.1.4 Model with GRU components for Manager and Worker

A new variant of the LeakGAN was designed newly for the research. Both the Manager and the Worker components are modified to be GRU-based models. The other parameters are as same as the base model. The results can be found in the Table 26.

8.4.1.5 Model with GRU components for Manager and Worker

A new variant of the LeakGAN was designed newly for the research. Both the Manager and the Worker components are modified to be GRU-based models. And a modification was made to train the discriminator. Many researchers state that the discriminator must not outperform the generator, as when that happens, the generator cannot learn well to generate new data. Therefore, during the pre-training phase, the discriminator is trained for 3 steps, and during the adversarial phase, the discriminator is trained for 2 steps. Other parameters were as same as the base model. The results can be found in the Table 26.

8.5 Benchmarking

As the baseline model is based on the LeakGAN, the results between the base model and the other variants of the LeakGAN model, which were created by the author, are compared. BLEU score is used to compare each model. Each model is separated based on the type of Manager component, type of Worker component, and the way how the discriminator is trained.

Model	Manager	Worker	Discriminator Training	BLEU Score
(Guo et al., 2018)	LSTM	LSTM	Pretraining (epoch = 3, step = 5) Adversarial training (epoch = 3, step = 5)	0.736

LeakGAN Variant 1	GRU	LSTM	Pretraining (epoch = 3, step = 5) Adversarial training (epoch = 3, step = 5)	0.759
LeakGAN Variant 2	LSTM	GRU	Pretraining (epoch = 3, step = 5) Adversarial training (epoch = 3, step = 5)	0.753
LeakGAN Variant 3	GRU	GRU	Pretraining (epoch = 3, step = 5) Adversarial training (epoch = 3, step = 5)	0.748
LeakGAN Variant 4	**GRU**	**GRU**	**Pretraining (epoch = 2, step = 3) Adversarial training (epoch = 2, step = 2)**	**0.836**

Table 26 - Evaluating results of variants of LeakGAN

Text generation is a deeply researched field in the field of computer science. There is a lot of research which can be compared to evaluate how the variants of the LeakGAN perform with respect to other architectures. The following is an evaluation of the models that were benchmarked using the COCO Image Captions dataset, and the BLEU metric was used to compare the models.

Work	BLEU Score
GenTex (The author's work)	**0.836**
RelGAN (Nie, Narodytska and Patel, 2018)	0.849
Meta-CoTGAN (Yin et al., 2020)	0.858
JSD-GAN (Li et al., 2019)	0.894
SeqGAN (Yu et al., 2017)	0.745
RankGAN (Lin et al., 2018)	0.743
LeakGAN (Guo et al., 2018)	0.746
IRL (Shi et al., 2018)	0.829
SAL (Zhou et al., 2020)	0.785

FGGAN (Yang et al., 2020)	0.773

Table 27 - Comparing variant of LeakGAN with other benchmarks

8.6 Functional Testing

The functional testing was done based on the functional requirements specified in the Software Requirement Specification chapter. The "Must Have" functionalities are implemented successfully.

Test Case	FR ID	User action	Expected Result	Actual Result	Result Status
1	FR1	The end-user **must** be able to upload text data to the system.	The dataset gets uploaded to the system	The dataset gets uploaded to the system	Passed
2	FR2	The end-user **must** be able to train the generative model by using the system.	The training process must start	The training process starts	Passed
3	FR3	The end-user **must** be able to export/download the generated data.	The generated text file must get downloaded	The generated text file gets downloaded	Passed

Table 28 - Functional Requirements Testing

8.7 Non-Functional Testing

Out of the three "Must Have" non-functional requirements, all three of them were successfully implemented.

Test Case	NFR ID	User action	Expected Result	Actual Result	Result Status

1	NFR1	The functionalities of the model **must** be easily visualized in the GUI.	The functionalities need to be easily visible.	The functionalities are simple and easily usable.	Passed
2	NFR2	The GUI **must** be able to run on a web browser (e.g., Chrome, Firefox, etc.).	Successfully running it on a browser	The GUI is running on a browser.	Passed
3	NFR3	The research **must** be easy to reproduce and extend on.	The core code files should be updated and easy to reproduce.	The code files are updated with the modified version of model.	Passed

Table 29 - Non-Functional Requirements Testing

8.8 Limitations of the testing process

Training a GAN is a resource extensive task, therefore both monetary and hardware resources were necessary. Monetary resource became critical, as each GAN model too around 24 hours to train. Therefore, more variants of the models were unable to be trained. And since the training is resource extensive, the number of epochs needed to train the model was reduced; however, the results during the training were comparable with the results mentioned in the paper, and the variants developed by the author outperformed the base model.

Due to lack of fund, only short text generation was done, that is working on the COCO Image Captions. The medium text generation, using the WMT 2017 dataset was not conducted.

BLEU metric was used to measure the quality of the generated text. However, ROUGE is currently a standard in measuring the quality of text. However, most of the previous works were done using BLEU score, so to benchmark the new models with existing works, BLEU score was used.

8.9 Chapter Summary

This chapter focused on testing the components of the research. The models that were created by the author were evaluated extensively based on the necessary metrics. Benchmarking was also done to evaluate where the current model stands when compared with other models. Then, the

testing for the functional and the non-functional requirements was discussed and finally, the limitations faced during the testing were stated.

CHAPTER 9: EVALUATION

9.1 Chapter Overview

This chapter focuses on evaluating the system after the core components of the system are implemented. The output that is obtained from the system is evaluated by two means: by industry experts and by self-evaluation.

9.2 Evaluation Methodology and Approach

Qualitative and quantitative evaluation methods were chosen to evaluate the research. Qualitative evaluation is done as a thematic analysis, where questions were constructed in a way where insights can be obtained from the research. Quantitative analysis is conducted by using metrics that were obtained after a thorough literature review, enabling the author to compare the research with other research in the text generation domain.

9.3 Evaluation Criteria

The following points were considered to evaluate the project in a qualitative way. These qualitative points are formulated based on the self-evaluation done by the author, and also by having interviews with experts. The following table depicts the criteria considered to conduct the qualitative analysis of the research.

Evaluation Criterion	**Purpose of Evaluation**
The choice of the problem domain, research gap, and its depth	This criterion is to verify whether the topic that was chosen for the research is valuable to the problem domain, and to make sure that the research is deep enough to solve the gap that was identified.
Research methodology and approach	This criterion focuses on evaluating the research approach and methodology that were taken to complete the research.

Literature review that was conducted	To evaluate how well the literature review was done to identify the existing systems and technologies that would help to solve the research gap.
The development of the core functionality	This criterion is to evaluate how well the implementation is done to bridge the research gap.
Evaluation of results and benchmarking	To evaluate the metrics used for the research, and how well it has performed compared to other past works in the same domain.

Table 30 - Evaluation Criteria

9.4 Self-Evaluation

Evaluation Criterion	**Evaluation remarks**
The choice of the problem domain, research gap, and its depth	Text generation is a domain which is used in many NLP tasks, therefore there are many opportunities to identify research gaps. The author had considered a problem in the open text generation, where text is generated without a context. Mostly, RNN based methods are used for generating text. However, they face issues during generation, and there are other alternatives which can produce quality textual data. The author feels that the research could have also focused on category-based text generation; however, being relatively new to the research domain, in the author's point of view, the research is a successful one, and would open to future enhancements in text generation.
Literature review and research methodology.	Text generation is a vast field, and there are many techniques that have been implemented. Therefore, a comprehensive literature survey is a must. A comprehensive survey was done to identify the research gaps, and then a review was done to identify the techniques on how to bridge the gap that was identified. The author feels that the literature review was completed successfully, and a proper research methodology was followed throughout the research.

The development of the core functionality	To bridge the research gap, an existing framework was utilized. Since a GAN based method was used, a GAN framework was researched and identified, and then modifications were made in the internal architecture to improve the results. The author feels that the development of the core was handled in an acceptable manner.
Evaluation of results and benchmarking	Many metrics are used to evaluate text generation. However, from the literature review that was conducted, BLEU and NLL were the most used metrics in much research, so using these would enable comparisons between previous works. Therefore, the research was benchmarked with other works. The author feels that the selected metrics adequately represent the research's outcomes.

Table 31 - Self-evaluation

9.5 Selection of the Evaluators

The research required evaluation from various personnel, who are experienced in the domain, to obtain insights from the conducted research. The evaluations were obtained from **2 lecturers**, **1 postdoctoral candidate**, **1 doctorate**, **4 industrial experts** and **3 graduates**.

9.6 Evaluation Result

Evaluation Criterion	**Summaries of Evaluation Remarks from Evaluators**
The choice of the problem domain, research gap, and its depth	The identified research gap is appreciable and relevant to the problem domain, text generation. The depth of the research was well done. However, research could have also focused on different types of generation and category-based generation.

	GANs are used extensively in image tasks and doing research on text tasks is interesting. The research was well conducted.
Literature review that was conducted	A proper survey and review have been conducted on the field of text generation. The author had managed to find many existing works on text generation and have collected evidence to support the gap. A large part of text generation domain is covered. More focus could have also been focused on the impact of Transformers on generating text.
Research methodology and approach	An acceptable research methodology was used to conduct the research. For an undergraduate level, the research done was good. The new state-of-the-art techniques, such as Transformers, could have been used to build components. But the research was done well, and the methodology used was consistent.
The development of the core functionality	The development is done well. Instead of coding from scratch, good research was done to select the framework. A good selection of development framework to start off with the development. The changes modified were executed well. Using GANs to generate text is good research, and the development of a complicated GAN model was handled well, with the selection of framework.
Evaluation of results and benchmarking	The metrics chosen to do the evaluation of the model was well researched. Proper literature review enabled the ability to benchmark the results with different works.

	Using BLEU is acceptable. However, ROUGE score is an improvement over BLEU, and that could have been considered. Benchmarking is done well. A good evaluation was done with 2 metrics. Although benchmarking was done from the results obtained using literature review, the framework can be used to perform benchmarking again, with the existing training data.

Table 32 - Evaluation Result

9.7 Limitations of Evaluation

There were a few limitations faced by the author to conduct the evaluations. Due to COVID-19 issues and the unrest situation in the country, face to face interviews and model demonstration was not conducted. Also, experts specific to GANs and text generation were limited, therefore feedback from experts and graduates related to deep learning were considered.

9.8 Evaluation on Functional Requirements

Out of the three functional requirements, all were implemented, and the results are displayed in appendix.

9.9 Evaluation on Non-Functional Requirements

Out of the three functional requirements, two were implemented, and the results are displayed in appendix.

9.10 Chapter Summary

The chapter presented a thorough discussion on the evaluation strategies used to evaluate the research. A thematic analysis was presented, with certain points on evaluating the quality of the

research. Based on the themes, questions were provided to experts for their opinions, and a self-evaluation from the author was obtained.

CHAPTER 10: CONCLUSION

10.1 Chapter Overview

The final chapter of the thesis, which focuses on the summary of the research. The goals of the research, the learning outcomes, and the research objectives are discussed. Also, the use of knowledge that was gained throughout the four academic years, the limitations faced during the research, and the future works that can arise from this research are discussed.

10.2 Achievements of Research Aims & Objectives

The aim of the research is to analyze, design, implement, and evaluate a text generation system that could successfully take a dataset of text sentences as input to generate meaningful and quality sentences.

The aim of the research was achieved, as a system to generate meaningful text was analyzed, designed, implemented, and evaluated. The following table represents the modules and how they have been useful for the author.

10.3 Utilization of Knowledge from the Course

The modules provided by University of Westminster, ranging from the first year to the final year, was highly beneficial in tackling the final year research.

Module	Description
Programming Principles I and II, Object-Oriented Programming	The Programming Principles module, which introduced Python and Programming, was insightful. The OOP concepts learned was helpful in identifying many components of the development framework, as OOP was used extensively.

Software Development Group Project	The module which introduced the author to the field of Data Science and AI, and the primary reason why such a research domain was chosen. This module assisted in identifying a research problem and the way to bridge the gap caused by the problem.
Algorithms: Theory, Design, and Implementation	Another important course that introduced the author to many algorithms and data structures. This was helpful in implementing useful algorithms and increased the critical thinking of the author.
Web Design and Development	Provided the foundational knowledge to the author for building web applications and GUIs.

Table 33 - Utilization of Knowledge from the Course

10.4 Use of Existing Skills

The author's existing skillset was helpful in tackling the initial part of the research. The following are some of the skills the author possessed before starting the research.

- The primary source of knowledge the user had on Data Science, NLP, and AI are thanks to online courses provided by Coursera. The Deep Learning Specialization, GANs specialization and the NLP specialization, which are provided by deeplearning.ai, were useful to get a grasp over the research problem.
- During the placement year, the author interned as a Data Scientist at Rootcode Labs, where the user got to work with cutting edge AI products. It is also important that the author was introduced to GANs during his stint at Rootcode Labs.

10.5 Use of New Skills

Throughout the year, as the research progressed, the author gained new skills, as the user was able to tackle the research problem.

- Utilizing GANs for language generation is a skill that the author did not possess before the research began, as there are no online courses nor videos. Therefore, as the author

conducted the research, and literature review, the user was able to learn a lot on GANs and text generation.

- The author also utilized open-sourced frameworks and existing codebases to modify the code and obtain results. The author states this as a skill, as he learned that rather than reinventing the wheel, enhancing it is efficient and effective.

10.6 Achievement of Learning Outcomes

Learnt by the author	Learning Outcome(s)
The author was able to learn on identifying a research gap successfully, identifying on how to solve the problem, and discovering the legal and ethical issues surrounding the problem.	LO4, LO6, LO2
The author was able to learn on how to carry out a successful literature review on many components of the research.	LO4, LO5
The author was able to learn on how to successfully conduct a software requirement analysis and identify the requirements of the project.	LO2, LO3, LO4
The author was able to learn on how to design a system meticulously, and to identify the components of the system.	LO1, LO2, LO3
The user was able to learn on building a prototype successfully and develop an evaluation process for the system.	LO1, LO7
The user was able to learn on successfully evaluating a system by using various methods and conducting benchmark testing to identify the performance of the research compared to existing works.	LO7, LO8

Table 34 - Achievement of Learning Outcomes

10.7 Problems and Challenges Faced

There were many challenges that were faced by the author when conducting the research. The following are the challenges and how they were handled.

Challenge/Problem faced	How it was solved
Finalizing a research problem	Finalizing the research problem was a challenging task faced by the author. The author had to spend around 3 months of research on various domains (Text to Image generation, Meme classification) and analysing their positives and negatives, and finally arrived at the problem of text generation. Consistent literature surveying assisted the author in solving this problem.
The massive scope of the research	The research on text generation is vast. There are multiple research works done ranging from decades. Again, the consistent surveying of literature assisted the author immensely in narrowing down the scope of the project.
Large learning curve	GANs and text generation are an unusual combination, therefore the author faced issues in learning. However, the literature and various open-sourced implementations was helpful in tackling the issue.
Long training times	Training a GAN for text generation takes a lot of time: the base model took more than 24 hours to train. Therefore, the author could not use free training resources such as Colab and Kaggle. Therefore, to solve this issue, the user had to use paid instances.
Monetary issues	Funds for training were crucial, as each model cost around $17 in training. The total cost ranged more than $100. The author had to conserve money to successfully conduct trainings. However, more iterations of models could not be trained as the cost had exceeded the intended budget.

Table 35 - Problems faced

10.8 Deviations

There were a few deviations faced by the author when conducting the research.

- The author's initial plan was to replace a RNN-based component with a Transformer component and monitor the results. However, the frameworks and open-sourced libraries

that were available were very limited in using Transformers. Therefore, the author opted to use memory components, such as LSTMs and GRUs.

- The author planned to train many variants with different parameters and components. However, as the cost exceeded, the author had to stop with five variants.

10.9 Limitations of the Research

There were few limitations of the research that was conducted.

- The cost spent for the model training is a major limitation, as it only allowed the user to train five different variants.
- The inability of many GAN-based frameworks to support Transformers.
- Lack of proper educative content on generating text with GANs.
- Lack of using the ROUGE metric for evaluation of generated text.

10.10 Future Enhancements

The research opens a plethora of possibilities that can be done in the future to enhance the research that was conducted.

- Modifying the training patterns of generator and discriminator of a GAN.
- Modifying the architecture of the discriminator.
- Replacing the RNN-based components with Transformers.
- Using ROUGE metric for evaluating the generated text.
- Attempting to use different RL training patterns.
- Attempting on using low-resource languages.

10.11 Achievement of the contribution to body of knowledge

The author was able to contribute to the problem domain in many ways:

- The author was able to successfully modify the Worker and Manager components of the selected LeakGAN model, train several variants, and obtain promising results.
- The author was able to modify the training pattern of the GAN (especially the discriminator) to obtain good results.

10.12 Concluding Remarks

The final chapter is a recollection of the author, of how the author had successfully completed the research aim and objectives that were set at the very early stage of the research, how the existing skills and new skills assisted the user, the challenges the user had to face and how he overcame them, the limitations that proved a hurdle for the user, and the future enhancements that can be conducted from the research. The research is very personal to the author as well, as the author is passionate towards GANs and NLP, and the author hopes that the readers would be able to gain useful insights from his research.

APPENDIX

Appendix 1 – References

Arjovsky, M., Chintala, S. and Bottou, L. (2017). *Wasserstein GAN*. [online] arXiv.org. Available at: https://arxiv.org/abs/1701.07875.

Bojar, O., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Edinburgh, J., Liu, Q., Monz, C., Negri, M., Post, M., Hopkins, J., Rubino, U., Specia, L. and Turchi, M. (2017). Findings of the 2017 Conference on Machine Translation (WMT17). *Proceedings of the Conference on Machine Translation (WMT)*, [online] 2, pp.169–214. Available at: https://aclanthology.org/W17-4717.pdf [Accessed 21 Oct. 2021].

Che, T., Li, Y., Zhang, R., Hjelm, R.D., Li, W., Song, Y. and Bengio, Y. (2017). Maximum-Likelihood Augmented Discrete Generative Adversarial Networks. *arXiv:1702.07983 [cs]*. [online] Available at: https://arxiv.org/abs/1702.07983 [Accessed 17 May 2022].

Chen, J., Wu, Y., Jia, C., Zheng, H. and Huang, G. (2019). Customizable text generation via conditional text generative adversarial network. *Neurocomputing*. doi:10.1016/j.neucom.2018.12.092.

Chen, X., Fang, H., Lin, T.-Y., Vedantam, R., Gupta, S., Dollar, P. and Zitnick, C.L. (2015). Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv:1504.00325 [cs]*. [online] Available at: https://arxiv.org/abs/1504.00325.

Chowdhury, M.F. (2014). Interpretivism in Aiding Our Understanding of the Contemporary Social World. *Open Journal of Philosophy*, [online] 04(03), pp.432–438. doi:10.4236/ojpp.2014.43047.

Chung, J., Gulcehre, C. and Cho, K. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. [online] Available at: https://arxiv.org/pdf/1412.3555.pdf.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. [online] arXiv.org. Available at: https://arxiv.org/abs/1412.3555.

de Rosa, G.H. and Papa, J.P. (2021). A survey on text generation using generative adversarial networks. *Pattern Recognition*, 119, p.108098. doi:10.1016/j.patcog.2021.108098.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. [online] arXiv.org. Available at: https://arxiv.org/abs/1810.04805.

Dubois, A. and Gadde, L.-E. (2002). Systematic combining: an abductive approach to case research. *Journal of Business Research*, [online] 55(7), pp.553–560. doi:10.1016/s0148-2963(00)00195-8.

Fedus, W., Goodfellow, I. and Dai, A.M. (2018). MaskGAN: Better Text Generation via Filling in the_____. *arXiv:1801.07736 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1801.07736.

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). *Generative Adversarial Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1406.2661.

Guimaraes, G.L., Sanchez-Lengeling, B., Outeiral, C., Farias, P.L.C. and Aspuru-Guzik, A. (2018). Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv:1705.10843 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1705.10843 [Accessed 21 Oct. 2021].

Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y. and Wang, J. (2018). Long Text Generation via Adversarial Training with Leaked Information. *Proceedings of the AAAI Conference on Artificial Intelligence*, [online] 32(1). Available at: https://ojs.aaai.org/index.php/AAAI/article/view/11957 [Accessed 21 Oct. 2021].

Hjelm, R.D., Jacob, A.P., Che, T., Trischler, A., Cho, K. and Bengio, Y. (2018). Boundary-Seeking Generative Adversarial Networks. *arXiv:1702.08431 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1702.08431 [Accessed 21 Oct. 2021].

Hochreiter, E., Rovelli, R. and Winckler, G. (1997). Centrální banky a využití ražebného? studie tří transformujících se ekonomik. *Politická ekonomie*, [online] 45(2), pp.193–206. doi:10.18267/j.polek.274.

Hochreiter, S. and Schmidhuber, J. (1997). Flat Minima. *Neural Computation*, [online] 9(1), pp.1–42. doi:10.1162/neco.1997.9.1.1.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, [online] 79(8), pp.2554–2558. doi:10.1073/pnas.79.8.2554.

House, E.R. (1991). Realism in Research. *Educational Researcher*, 20(6), pp.2–9. doi:10.3102/0013189x020006002.

Hyde, K.F. (2000). Recognising deductive processes in qualitative research. *Qualitative Market Research: An International Journal*, [online] 3(2), pp.82–90. doi:10.1108/13522750010322089.

Jang, E., Gu, S. and Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax. *arXiv:1611.01144 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1611.01144 [Accessed 21 Oct. 2021].

Kingma, D.P. and Welling, M. (2013). *Auto-Encoding Variational Bayes*. [online] arXiv.org. Available at: https://arxiv.org/abs/1312.6114.

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M. and Inman, D.J. (2019). 1D Convolutional Neural Networks and Applications: A Survey. *arXiv:1905.03554 [cs, eess]*. [online] Available at: https://arxiv.org/abs/1905.03554 [Accessed 13 Sep. 2020].

Kusner, M.J. and Hernández-Lobato, J.M. (2016). GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution. *arXiv:1611.04051 [cs, stat]*. [online] Available at: https://arxiv.org/abs/1611.04051 [Accessed 21 Oct. 2021].

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). *Lecun 98*. [online] Available at: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf.

Li, Y., Pan, Q., Wang, S., Yang, T. and Cambria, E. (2018). A Generative Model for category text generation. *Information Sciences*, [online] 450, pp.301–315. doi:10.1016/j.ins.2018.03.050.

Li, Z., Xia, T., Lou, X., Xu, K., Wang, S. and Xiao, J. (2019). *Adversarial Discrete Sequence Generation without Explicit NeuralNetworks as Discriminators*. [online] proceedings.mlr.press. Available at: http://proceedings.mlr.press/v89/li19g.html [Accessed 17 May 2022].

Lin, K., Li, D., He, X., Zhang, Z. and Sun, M.-T. (2018). Adversarial Ranking for Language Generation. *arXiv:1705.11001 [cs]*. [online] Available at: https://arxiv.org/abs/1705.11001 [Accessed 21 Oct. 2021].

Liu, D., Fu, J., Qu, Q. and Lv, J. (2019). BFGAN: Backward and Forward Generative Adversarial Networks for Lexically Constrained Sentence Generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12), pp.2350–2361. doi:10.1109/taslp.2019.2943018.

Liu, L. (2016). Using Generic Inductive Approach in Qualitative Educational Research: A Case Study Analysis. *Journal of Education and Learning*, [online] 5(2), pp.129–135. Available at: https://eric.ed.gov/?id=EJ1097415.

Mirza, M. and Osindero, S. (2014). *Conditional Generative Adversarial Nets*. [online] arXiv.org. Available at: https://arxiv.org/abs/1411.1784.

MIURA, K. (2011). An Introduction to Maximum Likelihood Estimation and Information Geometry. *Interdisciplinary Information Sciences*, 17(3), pp.155–174. doi:10.4036/iis.2011.155.

Morgan, D.L. (2014). Pragmatism as a Paradigm for Social Research. *Qualitative Inquiry*, 20(8), pp.1045–1053. doi:10.1177/1077800413513733.

Myung, I.J. (2003). Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1), pp.90–100. doi:10.1016/s0022-2496(02)00028-7.

Nagalavi, D. and Hanumanthappa, M. (2016). N-gram Word prediction language models to identify the sequence of article blocks in English e-newspapers. *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*. doi:10.1109/csitss.2016.7779376.

Nie, W., Narodytska, N. and Patel, A. (2018). *RelGAN: Relational Generative Adversarial Networks for Text Generation*. [online] openreview.net. Available at: https://openreview.net/forum?id=rJedV3R5tm [Accessed 21 Oct. 2021].

Nielsen, F. (2020). On a Generalization of the Jensen–Shannon Divergence and the Jensen–Shannon Centroid. *Entropy*, 22(2), p.221. doi:10.3390/e22020221.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002). *BLEU: a Method for Automatic Evaluation of Machine Translation*. [online] Available at: https://aclanthology.org/P02-1040.pdf.

Park, Y.S., Konge, L. and Artino, A.R. (2020). The Positivism Paradigm of Research. *Academic Medicine*, [online] 95(5), pp.690–694. doi:10.1097/acm.0000000000003093.

Pascanu, R., Mikolov, T. and Bengio, Y. (2013). *On the difficulty of training recurrent neural networks*. [online] Available at: https://proceedings.mlr.press/v28/pascanu13.pdf [Accessed 21 Oct. 2021].

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. [online] Available at: https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J. (2019). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. [online] arXiv.org. Available at: https://arxiv.org/abs/1910.10683.

Rizzo, G. and Van, T.H.M. (2020). Adversarial text generation with context adapted global knowledge and a self-attentive discriminator. *Information Processing & Management*, p.102217. doi:10.1016/j.ipm.2020.102217.

Shi, Z., Chen, X., Qiu, X. and Huang, X. (2018). Toward Diverse Text Generation with Inverse Reinforcement Learning. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. doi:10.24963/ijcai.2018/606.

statmt.org. (2017). *Translation Task - ACL 2017 Second Conference on Machine Translation*. [online] Available at: https://statmt.org/wmt17/translation-task.html [Accessed 3 Mar. 2022].

Sutskever, I., Vinyals, O. and Le, Q.V. (2014). *Sequence to Sequence Learning with Neural Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1409.3215.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, Aidan N, Kaiser, L. and Polosukhin, I. (2017). *Attention Is All You Need*. [online] arXiv.org. Available at: https://arxiv.org/abs/1706.03762.

Wang, H., Qin, Z. and Wan, T. (2018). Text Generation Based on Generative Adversarial Nets with Latent Variables. *Advances in Knowledge Discovery and Data Mining*, pp.92–103. doi:10.1007/978-3-319-93037-4_8.

Wang, K. and Wan, X. (2019). Automatic generation of sentimental texts via mixture adversarial networks. *Artificial Intelligence*, 275, pp.540–558. doi:10.1016/j.artint.2019.07.003.

WilliamLam (2022). *TextGAN-PyTorch*. [online] GitHub. Available at: https://github.com/williamSYSU/TextGAN-PyTorch [Accessed 3 Mar. 2022].

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4), pp.229–256. doi:10.1007/bf00992696.

Wu, Q., Li, L. and Yu, Z. (2021). TextGAIL: Generative Adversarial Imitation Learning for Text Generation. *arXiv:2004.13796 [cs]*. [online] Available at: https://arxiv.org/abs/2004.13796# [Accessed 2 Nov. 2021].

Xu, J., Ren, X., Lin, J. and Sun, X. (2018). Diversity-Promoting GAN: A Cross-Entropy Based Generative Adversarial Network for Diversified Text Generation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. doi:10.18653/v1/d18-1428.

Yang, S., Yu, X. and Zhou, Y. (2020). *LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example*. [online] IEEE Xplore. doi:10.1109/IWECAI50956.2020.00027.

Yang, Y., Dan, X., Qiu, X. and Gao, Z. (2020). FGGAN: Feature-Guiding Generative Adversarial Networks for Text Generation. *IEEE Access*, 8, pp.105217–105225. doi:10.1109/access.2020.2993928.

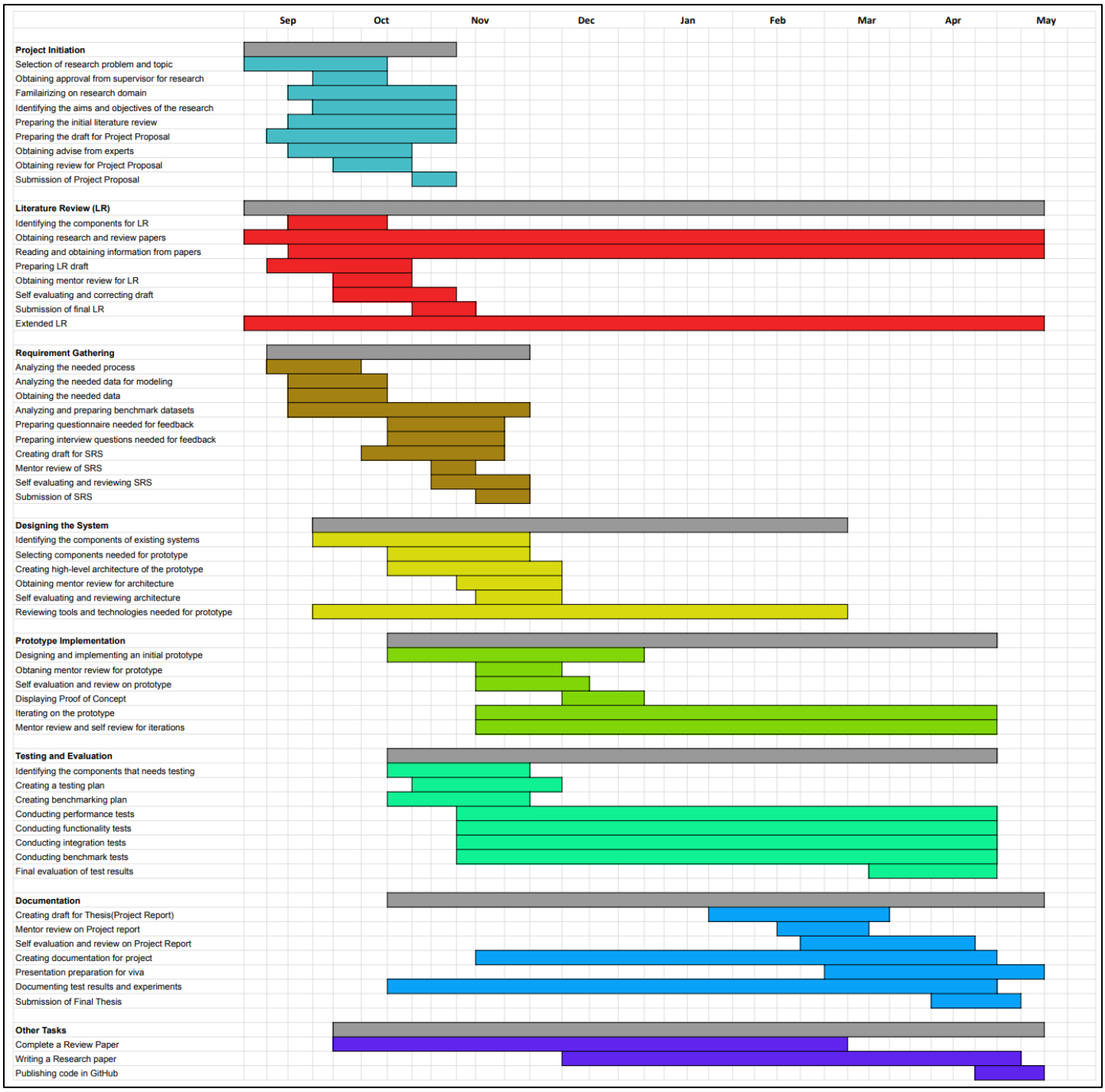
Yin, H., Li, D., Li, X. and Li, P. (2020). Meta-CoTGAN: A Meta Cooperative Training Paradigm for Improving Adversarial Text Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05), pp.9466–9473. doi:10.1609/aaai.v34i05.6490.

Yu, L., Zhang, W., Wang, J. and Yu, Y. (2017). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, [online] 31(1). Available at: https://ojs.aaai.org/index.php/AAAI/article/view/10804 [Accessed 21 Oct. 2021].

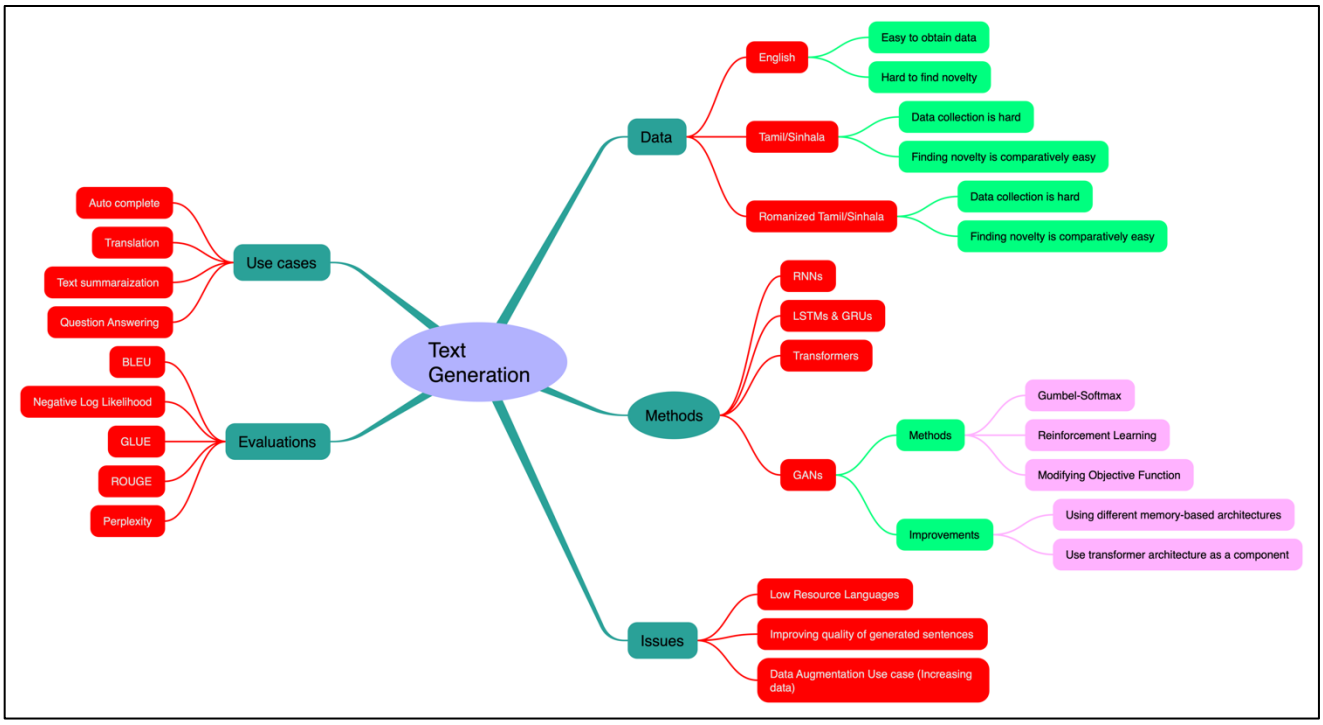
Zhang, R., Wang, Z., Yin, K. and Huang, Z. (2019). Emotional Text Generation Based on Cross-Domain Sentiment Transfer. *IEEE Access*, 7, pp.100081–100089. doi:10.1109/access.2019.2931036.

Zhou, W., Ge, T., Xu, K., Wei, F. and Zhou, M. (2020). Self-Adversarial Learning with Comparative Discrimination for Text Generation. *arXiv:2001.11691 [cs]*. [online] Available at: https://arxiv.org/abs/2001.11691 [Accessed 17 May 2022].

APPENDIX 2 – GANTT CHART



APPENDIX 3 – CONCEPT MAP



Appendix 4 – Questionnaire for Requirement Elicitation

A Survey on Language Modelling and Text Generation

With the renaissance of deep learning, Natural Language Processing (NLP) was one of the domains which displayed a huge boost in research, as neural networks enhanced many NLP applications: text classification, text-to-speech and speech-to-text facilities, information extraction, text generation, etc.

Language modelling has evolved exponentially in the past decades. It has enabled a machine to understand and generate text similar to a human being. This has been made possible by machine learning and deep learning. Artificial Neural Networks, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Long Short Term Memory (LSTMs), etc. are Deep Learning architectures that have aided in improved performance of machines in text processing tasks.

Language modelling is a task performed by the machine to predict the next word/character in a sentence/document by understanding the language (grammar, meaning of words, context) the sentence/document is written in. This can be used for many tasks such as generation of new text, text classification, summarization, etc. People have come across many applications of Language Models in their day to day lives such as auto-complete functions, chat bots, paraphrasing software, etc.

Despite the advancement, there is room for improvement in existing Language Models. It was observed that these models show poor results when processing long sentences or when using low resource languages such as Sinhala or Tamil (note: low resource languages are languages which have less number of data resources. Often these languages are comparatively less researched as well).

I, Varatharajah Vaseekaran, am a final year Software Engineering undergraduate at the University of Westminster, and I am conducting a research on language modelling and text generation. This questionnaire is conducted as part of my final year research to gather data on necessary requirements for successful completion of the research.

This questionnaire will take a few minutes to fill. No personal information will be recorded during this process. The responses provided by you will remain anonymous and the data collected will solely be used for academic purpose.

Your responses are of high value and will help immensely in assisting my research. My sincere appreciation for allocating few minutes of your time to respond to this questionnaire.

Thank you.

* Required

1. What describes your current status the best? *

Mark only one oval.

- ◯ School student
- ◯ Undergraduate
- ◯ Postgraduate (Masters, PhD, Post Doctoral, etc.)
- ◯ Industrial Professional
- ◯ Lecturer/Teacher
- ◯ Other: _____

2. What would be your level of experience, working on Natural Language Processing? *

Mark only one oval.

- ◯ No experience
- ◯ Personal projects
- ◯ Less than or equal to 1 year of industrial experience
- ◯ More than 1 year of industrial experience
- ◯ Other: _____

3. Have you used products that use language models? (e.g. autocomplete, paraphrasing software, language translation, etc.) *

Mark only one oval.

- ◯ Yes *Skip to question 4*
- ◯ Maybe *Skip to question 4*
- ◯ No *Skip to question 8*

Questions about products using language

Since you (may) have used products that are powered by language models, please answer the following questions.

models

4. What are the type of products you have come across? *

Check all that apply.

- [ ] Autocomplete in Mails
- [ ] Autocomplete/next word prediction in mobile phone keyboards
- [ ] Paraphrasing softwares
- [ ] Language Translation
- [ ] Other: _____

5. Products where you have seen/used language models *

Check all that apply.

- [ ] Mail Applications (e.g. Smart Compose in GMail)
- [ ] Mobile keyboards (e.g. Apple, Samsung, Google, etc.)
- [ ] Speech recognition softwares (e.g. Siri, Alexa, etc.)
- [ ] Translating softwares (e.g. Google Translate, Bing Translate, etc.)
- [ ] Search Engines (Understanding search terms and summarizing answers)
- [ ] Other: _____

6. What are the issues you have faced when using such products? *

Check all that apply.

- [ ] Incorrectly predicting the "next word"
- [ ] Incorrectly summarizing or paraphrasing text
- [ ] Incorrectly translating text
- [ ] Less support for Sinhala/Tamil
- [ ] Other: _____

7. Which aspect should be improved when using such products?

Check all that apply.

- [ ] Improving the accuracy of predicting the next word
- [ ] Improving the accuracy and understandability of paraphrased/summarized texts
- [ ] Having more support for Sinhala and Tamil and other low-resource languages

Other: [ ] _____

Skip to question 8

Questions on technical aspect of the research

Following are some research questions based on the technical aspect of the research. This section would aid in selecting the suitable tools and technology for conducting the research.

8. What method/technique would you prefer when building language models? *

Check all that apply.

- [ ] I haven't worked on building language models
- [ ] RNN-based architectures (RNNs, LSTMs, GRUs, etc.)
- [ ] Transformer architectures (Pre-trained models such as GPT-2, BART, T5, etc.)
- [ ] GANs
- [ ] Variational AutoEncoders (VAEs)
- [ ] Hybrid approach (Mixture of the above mentioned methods)

Other: [ ] _____

9. What language is preferable for building and training language models? *

Mark only one oval.

- ◯ I haven't worked on building language models
- ◯ Python
- ◯ R
- ◯ C++
- ◯ Java
- ◯ Other: _____

10. What deep learning framework is/are preferable for building and training language models? *

Check all that apply.

- ☐ I haven't used any deep learning framework
- ☐ PyTorch
- ☐ TensorFlow
- ☐ Keras
- ☐ Trax
- Other: ☐ _____

11. What are the libraries and tools that are useful for building and training language models? *

Check all that apply.

- [ ] I haven't worked on building language models
- [ ] NLTK (Natural Language Toolkit)
- [ ] Sci-Kit Learn
- [ ] HuggingFace Transformers
- [ ] Pandas
- [ ] Numpy
- [ ] Jax
- [ ] mlr (R programming)
- [ ] Other: _____

12. What environment do you prefer when training deep learning models? *

Mark only one oval.

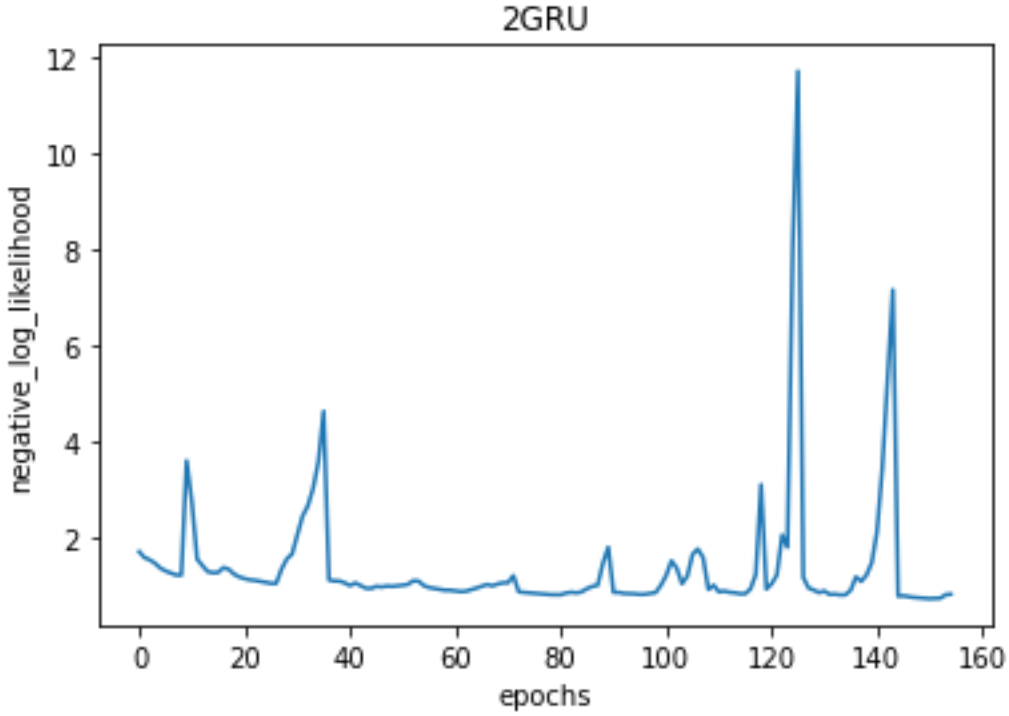
- ◯ I haven't worked on building deep learning models
- ◯ Own machine (with CPU and GPU)
- ◯ Cloud environment
- ◯ Depends on the data and task
- ◯ Other: _____

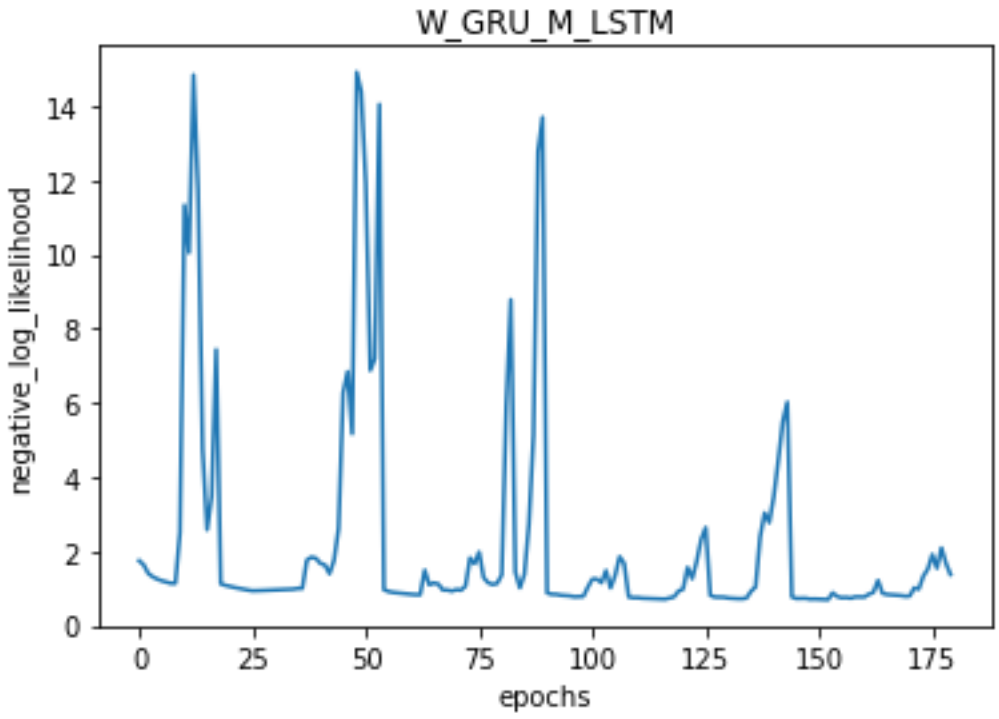
13. If you have worked on cloud environments or notebooks, what is/are the preferred options? *

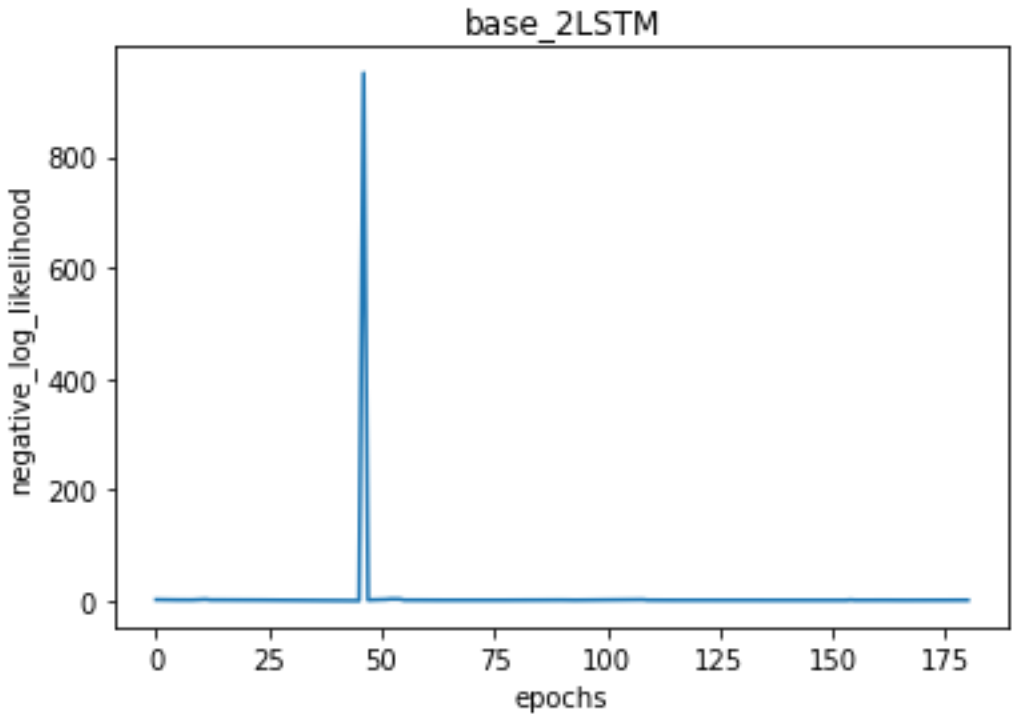
Check all that apply.

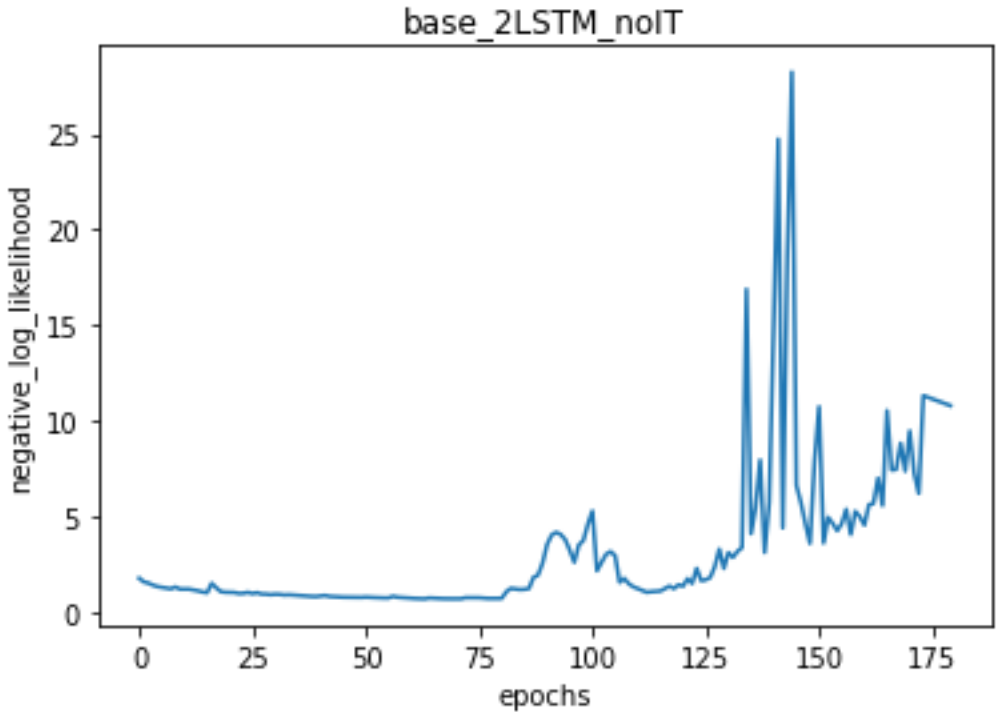
- [ ] I haven't used any cloud environments nor notebooks before
- [ ] Google Colaboratory
- [ ] Kaggle Notebook
- [ ] Paperspace Gradient
- [ ] AWS Sagemaker
- [ ] GCP Compute Engine
- [ ] GCP Vertex AI
- [ ] IBM Watson Studio
- [ ] Azure Notebooks
- [ ] Digital Ocean
- [ ] Other: _____

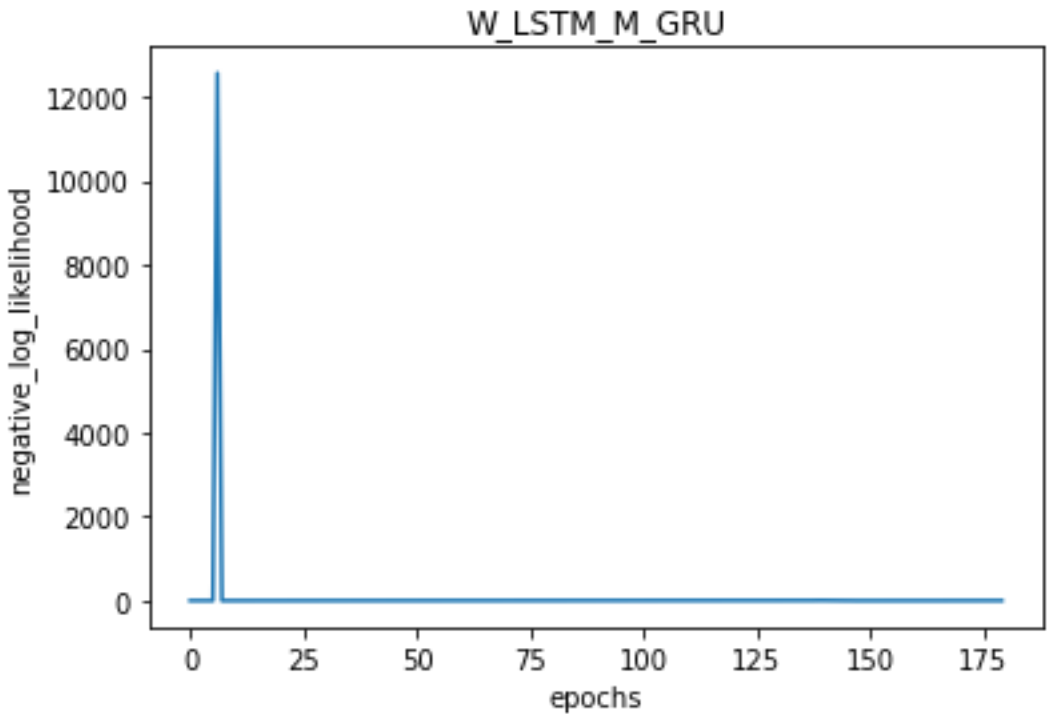
APPENDIX 5 – Negative Log-Likelihood Performances of Model

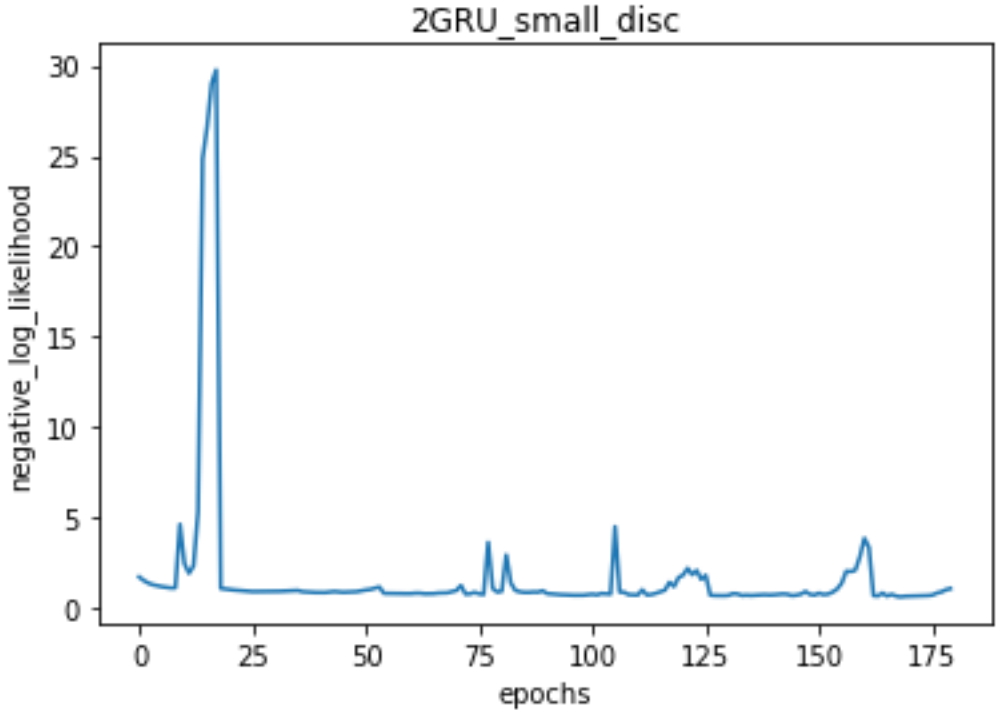




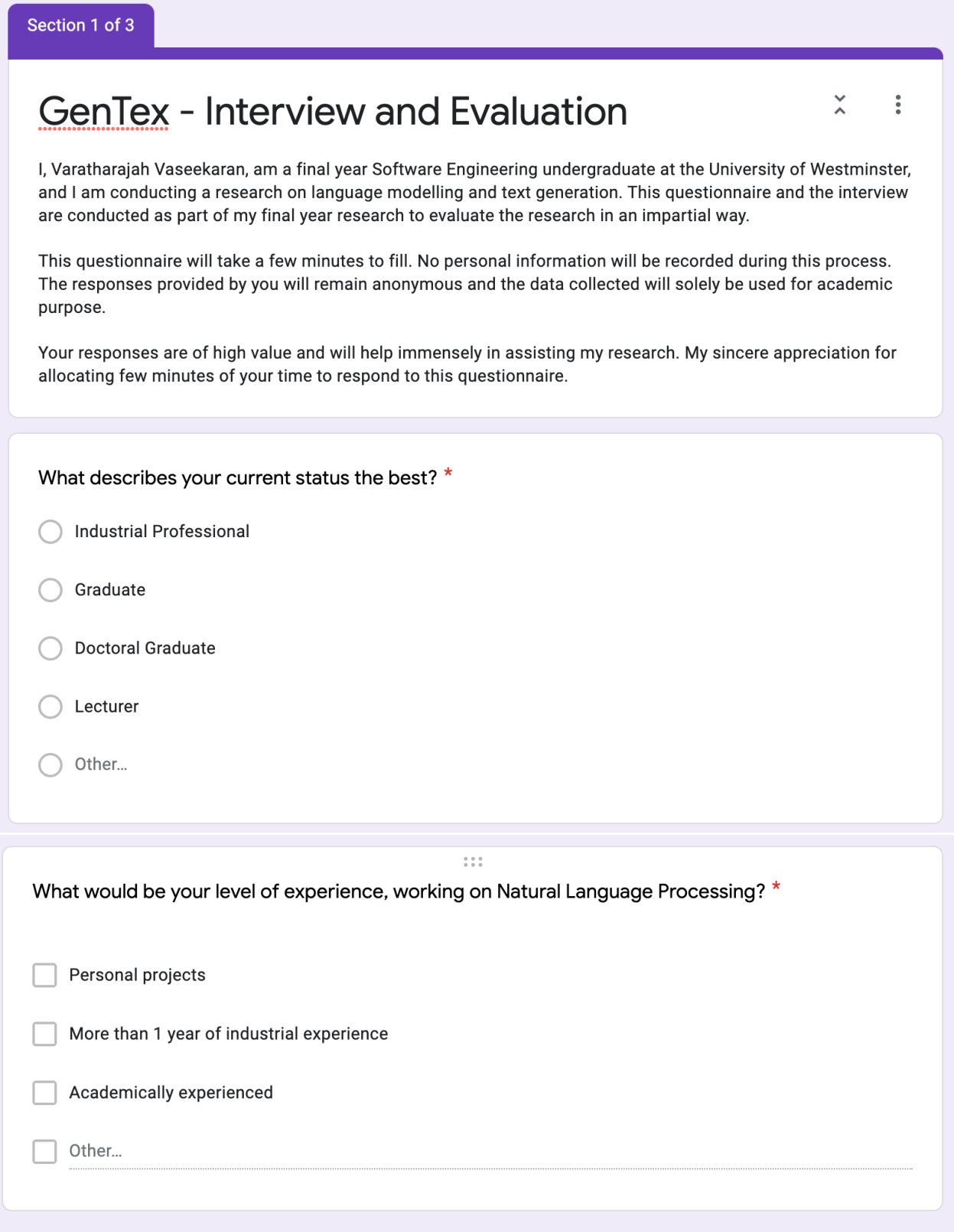

base_2LSTM
negative_log_likelihood
800
600
400
200
0
0
25
50
75
100
125
150
175
epochs


base_2LSTM_noIT
negative_log_likelihood
25
20
15
10
5
0
0
25
50
75
100
125
150
175
epochs


W_LSTM_M_GRU
negative_log_likelihood
epochs
0
2000
4000
6000
8000
10000
12000
0
25
50
75
100
125
150
175


2GRU_small_disc
negative_log_likelihood
epochs
0
5
10
15
20
25
30
0
25
50
75
100
125
150
175

APPENDIX 6 – Evaluation Form



Section 1 of 3

GenTex - Interview and Evaluation

I, Varatharajah Vaseekaran, am a final year Software Engineering undergraduate at the University of Westminster, and I am conducting a research on language modelling and text generation. This questionnaire and the interview are conducted as part of my final year research to evaluate the research in an impartial way.

This questionnaire will take a few minutes to fill. No personal information will be recorded during this process. The responses provided by you will remain anonymous and the data collected will solely be used for academic purpose.

Your responses are of high value and will help immensely in assisting my research. My sincere appreciation for allocating few minutes of your time to respond to this questionnaire.

What describes your current status the best? *

- Industrial Professional
- Graduate
- Doctoral Graduate
- Lecturer
- Other...

What would be your level of experience, working on Natural Language Processing? *

- Personal projects
- More than 1 year of industrial experience
- Academically experienced
- Other...

Evaluating the generated data

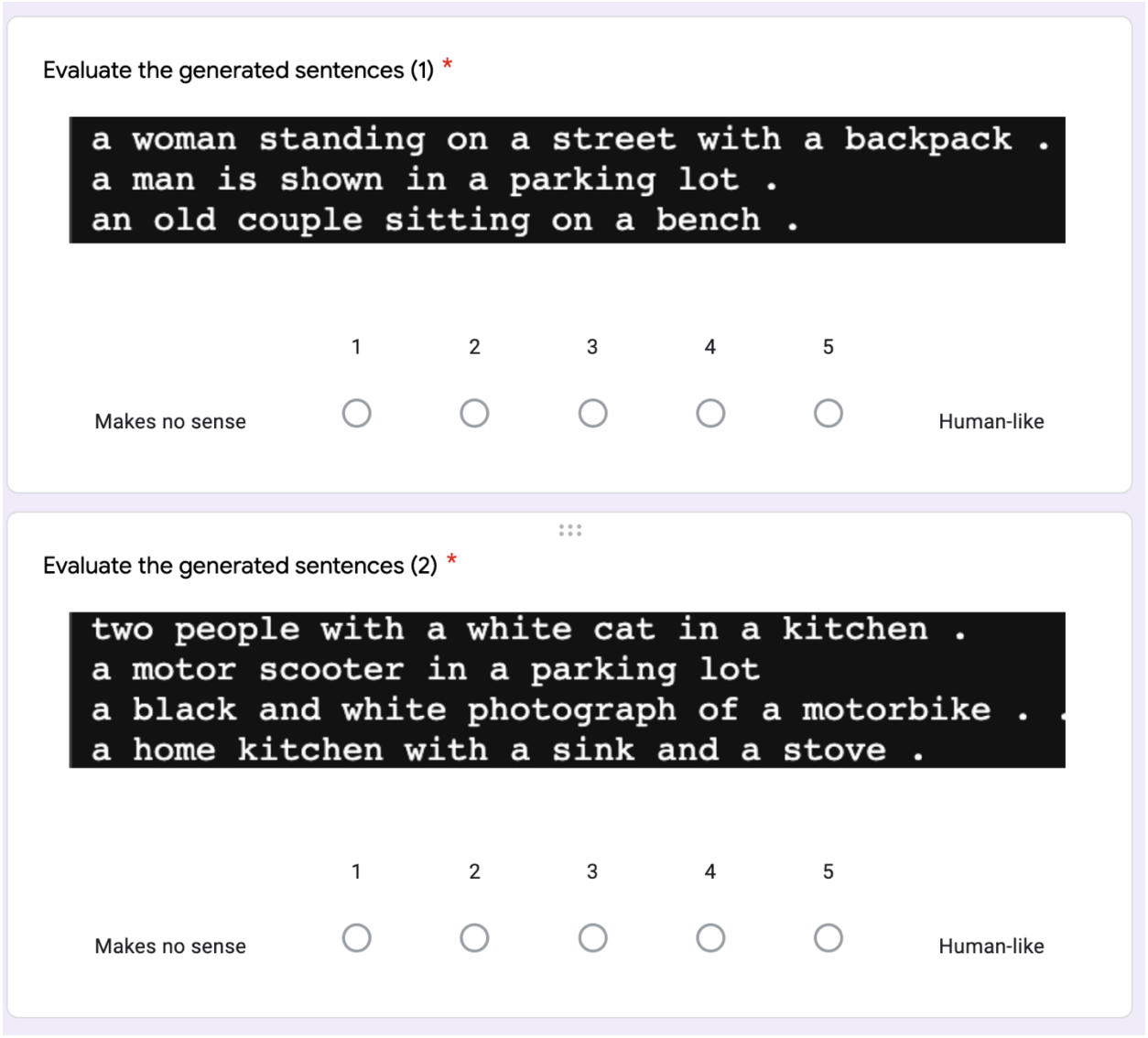
The following section is based on evaluating the generated data from the research, and to check how close is to human-level meaning.

Each question would have some sentences generated by the model. Please select a value in the scale (from 1 to 5), where 1 states the generated sentence is very poor, and 5 states that the generated sentence is human-like generation.

Note: the research was done using the COCO Image Captions data. A snapshot of the data would be provided for you to get an understanding the nature of the dataset.

Some examples of human written texts from the COCO Image Captions dataset

```
icycle replica with a clock as the front wheel.
lack honda motorcycle parked in front of a garage.
a room with blue walls and a white sink and door.
a car that seems to be parked illegally behind a legally parked car
a large passenger airplane flying through the air.
there is a gol plane taking off in a partly cloudy sky.
blue and white color scheme in a small bathroom.
this is a blue and white bathroom with a wall sink and a lifesaver on the wall.
a blue boat themed bathroom with a life preserver on the wall
the bike has a clock as a tire.
a honda motorcycle parked in a grass driveway
two cars parked on the sidewalk on the street
an airplane that is, either, landing or just taking off.
a bathroom with walls that are painted baby blue.
a bathroom with a toilet, sink, and shower.
a long empty, minimal modern skylit home kitchen.
an office cubicle with four different types of computers.
a bathroom sink with toiletries on the counter.
a small closed toilet in a cramped space.
two women waiting at a bench next to a street.
a bathroom sink and various personal hygiene items.
this is an open box containing four cucumbers.
an old-fashioned green station wagon is parked on a shady driveway.
a gas stove next to a stainless steel kitchen sink and countertop.
a black metal bicycle with a clock inside the front wheel.
a black honda motorcycle with a dark burgundy seat.
a tan toilet and sink combination in a small room.
several motorcycles riding down the road in formation.
a black cat is inside a white toilet.
city street with parked cars and a bench.
the home office space seems to be very cluttered.
a man in a wheelchair and another sitting on a bench that is overlooking the water.
a man sits with a traditionally decorated cow
rows of motor bikes and helmets in a city
a cute kitten is sitting in a dish on a table.
a messy bathroom countertop perched atop black cabinetry.
a man getting a drink from a water fountain that is a toilet.
an open food container box with four unknown food items.
an old teal colored car parked on the street.
```


Evaluate the generated sentences (1) *
a woman standing on a street with a backpack .
a man is shown in a parking lot .
an old couple sitting on a bench .
1
2
3
4
5
Makes no sense
Human-like
Evaluate the generated sentences (2) *
two people with a white cat in a kitchen .
a motor scooter in a parking lot
a black and white photograph of a motorbike .
a home kitchen with a sink and a stove .
1
2
3
4
5
Makes no sense
Human-like

Evaluation of the research

The final section of the evaluation focuses on the metrics obtained from the research, and also a critical review of the research conducted, stating any flaws and improvements.

Following is a benchmarking score conducted by (de Rosa and Papa, 2021). The current research focuses on the COCO Captions, and the LeakGAN is obtained as the base model for the research.

Table 2
BLEU-2 benchmark regarding architectures that use such a metric to evaluate the four most common datasets.

Architecture	Amazon Review	Chinese Poems	COCO Captions	WMT News
MLE	-	0.667	0.781	0.768
RNNLM	0.848	-	-	-
RelGAN	-	-	0.849	0.881
Meta-CoTGAN	-	-	0.858	0.882
SeqGAN	0.856	0.739	0.745	0.777
VGAN	0.868	-	-	-
RankGAN	-	0.812	0.743	0.727
LeakGAN	-	**0.881**	0.746	0.826
IRL	-	-	0.829	-
BFGAN	**0.920**	-	-	-
SAL	-	-	0.785	0.788
FGGAN	-	-	0.773	-
MaliGAN	-	0.741	-	-
JSD-GAN	-	0.536	**0.894**	**0.943**

What are the best metrics for evaluating machine generated text?

- [ ] BLEU (BiLingual Evaluation Understudy)
- [ ] ROUGE (Recall-Oriented Understudy for Gisting Evaluation)
- [ ] Perplexity
- [ ] NLL (Negative Log-Likelihood)
- [ ] Other...

The BLEU-2 metric that was obtained after training a variant of the LeakGAN is 0.836. Do you think the conducted research generated data better than the base model? *

- ( ) Yes
- ( ) No
- ( ) Maybe

The training time taken for the variant of LeakGAN is around 15 hours, whereas the training time for the base LeakGAN was a little over 24 hours. What is your view regarding this statement? *

- ○ The variant of the LeakGAN has outperformed the base model in both quality and speed
- ○ The speed of the training is not important
- ○ Other...

Any other techniques/architectures that could have improved the quality of the generated text? *

- ☐ RNN-based models
- ☐ LSTMs (Long Short Term Memory)
- ☐ GRUs (Gated Recurrent Units)
- ☐ Transformers (GPT-2)
- ☐ Other GAN-based models
- ☐ Reinforcement Learning
- ☐ Other...