How to tell Real From Fake? Understanding how to classify human-authored and machine-generated text

Debashish Chakraborty

Australian National University

A thesis submitted in partial fulfilment for the degree of

Master of Computing (Advanced) at
Research School of Computer Science
The Australian National University

May 2019

Except where otherwise indicated, this thesis is my own original work.

Debashish Chakraborty
30 May 2019

Acknowledgments

First, I would like to express my gratitude to Dr Patrik Haslum for his support and guidance throughout the thesis, and for the useful feedback from my late drafts.

Thank you to my parents and Sandra for their continuous support and patience. A special thanks to Sandra for reading my very "drafty" drafts.

Abstract

Natural Language Generation (NLG) using *Generative Adversarial Network*s (GANs) has been an active field of research as it alleviates restrictions in conventional *Language Modelling* based text generators e.g. Long-Short Term Memory (LSTM) networks. The adequacy of a GAN-based text generator depends on its capacity to classify human-written (real) and machine-generated (synthetic) text. However, traditional evaluation metrics used by these generators cannot effectively capture classification features in NLG tasks, such as *creative writing*. We prove this by using an LSTM network to almost perfectly classify sentences generated by a *LeakGAN*, a state-of-the-art GAN for long text generation.

This thesis attempts a rare approach to understand real and synthetic sentences using meaningful and interpretable features of long sentences (with at least 20 words). We analyse novelty and diversity features of real and synthetic sentences, generate by a *LeakGAN*, using three *meaningful* text dissimilarity functions: Jaccard Distance (JD), Normalised Levenshtein Distance (NLD) and Word Mover's Distance (WMD). In particular, these functions focus on (1) the number of common words, (2) the order of these words, and (3) the semantic similarity in both sentence types, making them interpretable. We provide a comprehensive investigation to identify the effectiveness of novelty and diversity, in classifying real and synthetic sentences, by training two different classification algorithms of varying complexities. Our evaluations show that sentence diversities, using JD and NLD, are the most effective features for classification of human-authored and machine-generated sentences.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural Language Generation (NLG) can be defined as the process of constructing natural language text or speech in English or any other language [Reiter and Dale, 2000]. One of the instances of NLG, *Neural Text Generation* (NTG), involves the task of generating text using *artificial neural networks*[1] and their variants. This task can be referred to as *text-to-text* generation when the process of text generation uses existing text as input, usually in a supervised setting. The goal of this task is to learn the distribution of text in the given text corpus to achieve some communicative goals in practical applications. Neural text generation has been able to achieve outstanding outcomes in the past few years [Wu et al., 2016; Hassan et al., 2018]. Some applications of such text generation can be seen in

- Creative writing e.g. storytelling, poetry-generation [Ghazvininejad et al., 2016]
- Machine translation, from one language to another [Yang et al., 2017; Wu et al., 2016]
- Abstractive summarisation, reduce long documents of text into short summaries while capturing and preserving the important details and meaning of the documents [Lloret and Palomar, 2012; Khatri et al., 2018] e.g. summarising news articles, scientific journals
- Freeform question answering: answer is generated, not extracted from a knowledge base [Choi et al., 2018]

Metrics or methodology on how to *automatically*[2] evaluate such text generation are still open research questions [Theis et al., 2016; Semeniuta et al., 2018]. The main challenge affecting current evaluation metrics is that they are not able to measure *sample diversity* and *sample quality* simultaneously. Sample diversity applies to a measure of non-repetitiveness in and between samples, while sample quality refers to notions like *adequacy* (is the text ambiguous or unclear), *accuracy* (does the text convey the information it is supposed to) and *fluency* (does the text present the information grammatically in a readable manner) [Dale and Mellish, 1998].

[1] Artificial neural networks can be defined as "a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." [Caudill, 1987]

[2] with little or no human input

A lot of research has been conducted towards evaluating *language model*s that generate text, and how this generated text differs from the text generated by other models [Semeniuta et al., 2018; Theis et al., 2016]. Most automatic evaluation metrics for generation tasks, like **machine translation**, indirectly evaluate sample quality by comparing the generated text to one or more human-written *reference* sentences [Papineni et al., 2002; Banerjee and Lavie, 2005; Lin, 2004], providing a degree of accuracy for these generated text. These methods, however effective for certain tasks, cannot be generalised across different applications of NTG. This is particularly true in the case of **creative writing**, where text generation occurs with no reference text to compare sample correctness. Current studies do not focus heavily on comparing the generated text with the text used to train language models, which may provide clues towards bridging the gap between the two.

This thesis focuses on discovering features of text, in this case sentences, to assist in the classification of *human-authored text* and *machine-generated text*. We refer to human-authored sentences as *real* sentences and machine-generated sentences as *synthetic* or fake sentences. We attempt to discover these features by comparing dissimilarity of generated sentences with each other as well as the real sentences used to generate the synthetic ones. The metrics for dissimilarity that will be explored in this thesis are

- *Novelty* of a sentence, a *between-set* metric, to investigate the dissimilarity of a sentence in a set of sentences with the sentences in a different set, and
- *Diversity* of a sentence, a *within-set* metric, to investigate the dissimilarity of a sentence in a set of sentences with the rest of sentences in the same set.

Using these features, we can build classifiers to differentiate real and synthetic sentences and potentially help us understand how to generate text that more closely resemble real text. Moreover, we concentrate on longer sentences with at least 20 words, which may present more pronounced features compared to shorter sentences.

1.1 Review of Neural Text Generation and recent advancements

The general approach to Neural Text Generation (NTG) is using a *probabilistic language modelling* approach, proposed by Bengio et al. [2003] as *neural probabilistic language model* (NNLM). These language models are trained by iterating over a sequence of tokens (character or words) *sequentially* from left from right. The tokens are sampled from a distribution conditioned on preceding words and a representation of the tokens generated until that iteration. These neural probabilistic language models can be viewed as an extension of the *n-gram*[3] language modelling paradigm which uses the *Markov assumption*, that the probability of a word depends only on *n-1* previous words.

[3] An n-gram is a sequence of *n* words e.g. a 2-gram, or bigram, is a sequence of two words.

The probability of a sequence of N words $\{w_1, \ldots, w_N\}$, in a probabilistic language model, can be denoted as $P(w_1, \ldots, w_N)$:

$$P(w_1, \ldots, w_N) = \prod_{i=1}^{N} P(w_i | w_1, \ldots, w_{i-1}) \tag{1.1}$$

This probability is generally conditioned on a window of n previous words instead of all previous words as the number of words occurring prior to a word w_i varies depending on its location in a document. So the probability can be re-written as

$$P(w_1, \ldots, w_N) = \prod_{i=1}^{N} P(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{N} P(w_i | w_{i-n}, \ldots, w_{i-1}) \tag{1.2}$$

Inherently, text contains long-distance dependencies, where an event currently being discussed may be related to event(s) that were explored much earlier in the text. NNLMs cannot express these dependencies as a consequence of Markov assumption, and therefore, fails to capture long-term dependencies itself [Rosenfeld, 2000].

1.1.1 Text generation using recurrent neural language models

In order to address the issue of long-term dependencies, *recurrent neural language model*s [Mikolov et al., 2010] were developed to encode previous *variant-length* tokens into a *hidden* vector, which is then used to predict the next word. These models use *Recurrent Neural Network*s (RNNs), which are the most widespread model for generating sequences. Most RNN language models adopt *Maximum Likelihood Estimation* (MLE) as their training objective, also known as *teacher forcing*, where the human-written words from sentences are used as input into the model, to be conditioned on, for generating subsequent words of a sentence. This approach suffers from *exposure bias* which causes problems when the model is frequently forced to condition on sequences of words during sentence generation that were never conditioned on at training time.

Text generation using RNN language models has been a field of active research. For example, improved variants of RNN, e.g. long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and gated recurrent unit (GRU) [Cho et al., 2014], show satisfactory results in capturing long-term dependencies in the text. These models have also demonstrated outstanding results in language modelling [Graves, 2013], machine translation [Yang et al., 2017; Wu et al., 2016], text classification [Miayto et al., 2016] etc. However, as pointed out by Bengio et al. [2015], replicating the distribution of previously seen text, due to the aforementioned exposure bias, does not imply generation of satisfactory text . In an attempt to decrease exposure bias, Bengio et al. [2015] proposed *Scheduled Sampling*, where randomness is introduced during each step of training to determine whether the model generates freely or performs teacher forcing. However, this was proven to be inconsistent by Husz'ar [2015].

1.1.2 Text generation using Reinforcement Learning with Adversarial Training

Some recent developments in the field of text generation [Yu et al., 2016; Guo et al., 2017] have tackled text generation using *Reinforcement Learning* (RL) [Sutton and Barto, 1998] and *Generative Adversarial Networks* (GANs) [Goodfellow et al., 2014]. RL is based on the concept of having an agent acting in an environment, *i.e.* a state space. Each action in the state space corresponds to a reward, and the objective of the agent is to take actions that maximise its cumulative reward based on its state. In order to decide which actions to take, the agent uses a policy, which recommends certain actions based on the state of the agent. GANs are frameworks for training generative models in an adversarial setup, with a *generator* generating samples that will try to fool a *discriminator* trained to differentiate real and synthetic samples. One limitation of GAN in text generation is that they are designed to generate continuous, real-valued data, such as images, while text consists of discrete tokens (words, letters, punctuation etc).

SeqGAN, proposed by Yu et al. [2016], is an attempt at generating text with a GAN which addresses the issue of the discrete nature of text by treating the generator as an agent in reinforcement learning. Research into extending GAN training to discrete samples has been a highly active area of research leading to the advent of architectures like TextGAN [Zhang et al., 2017], RankGAN [Lin et al., 2017], MaskGAN [Fedus et al., 2018], etc. However, the results from these models are limited to short generated text samples that are less than 20 words.

LeakGAN [Guo et al., 2017], which is adapted from SeqGAN, focuses on the generation of longer sentences, with at least 20 words. This model introduces a method to *leak* the discrimination criteria between the discriminator to the generator using advanced reinforcement learning methods [Vezhnevets et al., 2017]. The proposed model, which outperforms SeqGAN when the generated sentences are longer, is a robust framework for generation of long text. As we are more concerned about discovering features in longer sentences, we will be using LeakGAN as our means of generating synthetic text. Further details about the internal architecture of LeakGAN is discussed in Section 2.3.1.

1.2 Limitations of current evaluation techniques for machine-generated text

The current evaluation mechanism used during text generation [Yu et al., 2016; Guo et al., 2017] is primarily based on metrics derived from n-gram matching which are used to assess quality of generated samples e.g. BLEU, self-BLEU, n-BLEU etc. The BLEU [Papineni et al., 2002] metric is based on n-gram overlaps of tokens between *candidate* and *reference* sequences and how well the length of the candidate text matches the length of the reference. Semeniuta et al. [2018] demonstrate that these metrics are ineffective measures for evaluation of certain text generation applications such as creative writing, since these tasks do not have reference text for comparison.

Another common evaluation mechanism focuses on measuring the quality of a model independent of the application or the task, also known as intrinsic evaluation. A common measure in such evaluations often used to evaluate text generators [Fedus et al., 2018; Yu et al., 2016] is perplexity. This metric is based on the notion that the best language model is one that best predicts unseen sequences not used when training this model. However, perplexity is a feature of the language model and does not provide useful information about the generated text. Since the thesis focuses on extracting features from the generated text in order to classify them against human-written text, we do not include perplexity as one of our evaluation metrics.

Human evaluation has also been used to assess text generation, e.g. [Fedus et al., 2018; Weili Nie and Patel, 2019], and is often viewed as the gold standard evaluation. Although the ideal scenario would be to develop models that generate text which cannot be detected by humans as being machine-generated, there are several drawbacks of human evaluation. In comparison to automatic evaluation, human evaluation tends to be slower and generally more expensive. This becomes more evident with increase in the size of the data to be evaluated. Moreover, human evaluation is able to capture quality of text generation but cannot capture diversity. As an example, a language model that directly copies sentences from the training set would be accepted by humans to produce sentences of adequate quality. However, due to repetitiveness in the generated samples, the model generation would have inadequate diversity.

Automatic evaluation metrics, which are able to capture the dissimilarity between feature distributions of real and generated text, may facilitate progress of GAN-based text generation. These metrics can be potentially useful as features in tuning model parameters during the training stage. For example, LeakGAN discriminator can be *potentially* to *leak* these useful features to the generator. Our aim in this thesis to explore various metrics to gain insight into how generated text may differ from original text to discover a potential in the future for bridging the gap between the two forms of text.

1.3 Thesis Outline

The body of this thesis is divided into the following chapters:

Chapter 2 - Provides the necessary background on the architectures used in this thesis for text generation and metrics for the evaluation of human-authored and machine generated text. The background for classification models and metrics used for evaluating their performance is also presented.

Chapter 3 - Provides details about the experimental setup, including the dataset used along with any preprocessing required.

Chapter 4 - Presents the results, analysis and discussion of evaluation metrics for

human-authored and machine-generated text, as well the performance of classification models trained using these metrics. Some limitations of the current experimental setup are also discussed.

Chapter 5 - Provides a conclusion for this thesis and summarises the findings and results. Directions for future work are also suggested.

Appendix A - Presents samples of text from the EMNLP2017 WMT Dataset and text generated by LeakGAN.

Chapter 2

Background on Architectures and Evaluation Metrics

This chapter provides the theory required to explain the models used in the generation of synthetic text, and classification of human-authored and synthetic text. We also discuss the metrics used to obtain text features and the evaluation measures applied to determine performance of the classification models. The reader is assumed to have some prior knowledge of artificial neural networks and basic statistics and probability. In each section, we only provide the background necessary to understand the experimental setup and results of this thesis. We also direct the reader to reference background, where relevant.

Section 2.1 describes basic text processing techniques applied in this thesis: tokenisation and word embeddings.

Section 2.2 presents background on neural networks that are commonly used to generate text: Feedforward Neural Network, Recurrent Neural Network and Long Short-Term Memory network.

Section 2.3 introduces LeakGAN, the chosen architecture used in this thesis to generate synthetic text. It also provides the motivation for choosing LeakGAN in comparison to other available GAN based text generators.

Section 2.4 describes the metrics used to quantitatively evaluate both real and synthetic text.

Section 2.5 describes the functions used to calculate dissimilarity between two text bodies.

Section 2.6 provides background on the classification models used to classify real and synthetic text.

Section 2.7 discusses measures used in this thesis to determine the performance

of the classifiers during the classification of human-written and machine-generated text.

2.1 Text Processing

In order to use a given document or sentence in NLG architectures, we need to represent words as *structured* input to these models. This section discusses the major text processing techniques applied in this thesis.

2.1.1 Tokenisation

In order to process text, a crucial component is the identification of basic units of a given text, *tokens*, that form *sequences* when grouped together. The process of dividing a text or sequence into these tokens is called *tokenisation* [Manning et al., 2008]. The total set of tokens form a *vocabulary*. In this thesis, sequences correspond to sentences and the tokens almost exclusively correspond to words, with a few exceptions such as punctuation.

2.1.2 Word Embeddings

In order to gain the notion of similarity and differences between word tokens, they can be mapped into word vectors, which can then be compared using distance measures like Jaccard, Cosine etc. *Word embeddings* are representations of words as low-dimensional vectors in which similar words are expected to be close in the vector space [Collobert and Weston, 2008], words in this representations are referred to as *embedded words*. As an example, given two sentences and their respective word embeddings w_1 and w_2, cosine similarity can provide an numerical understanding of their similarity by measuring the cosine of angle between two vectors.

$$d(w_1, w_2) = \frac{w_1 \cdot w_2}{\|w_1\|_2 \|w_2\|_2} \tag{2.1}$$

We use pre-trained word embeddings, *word2vec* [Mikolov et al., 2013a] to calculate one of our text dissimilarity functions, Word Mover's Distance, which will be discussed in Section 2.5.3.

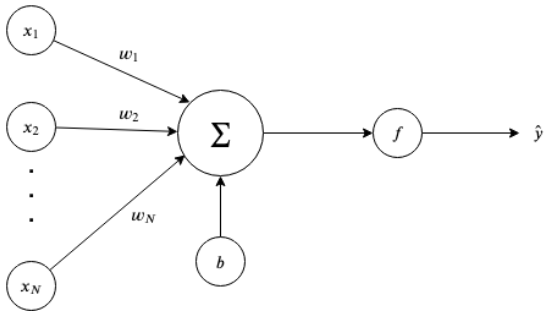

Figure 2.1: An artificial neuron with inputs $x_1, x_2, \ldots, x_N$, weights $w_1, w_2, \ldots, w_N$, bias b, activation function h and output $\hat{y}$.

2.2 Foundations of Neural Text Generation

As discussed in Chapter 1, Neural Text Generation (NTG), involves the task of generating text using artificial neural networks, generally referred to as neural networks and their variants. We also mentioned supervised text generation where one of the goals is to learn the distribution of text in a given text corpus. In this section, we will discuss the basic structure of different variants of neural networks and gain intuition on how they are able to learn an input distribution. Advanced variants, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, have also been examined. However, the internal details and mathematical background employed during the configuration of these networks are not essential for this thesis and will not be discussed. If the reader is interested in more information on these architectures, please refer to the *Deep Learning* book by Goodfellow et al. [2016].

2.2.1 Neural Networks

The most basic and widely used artificial neural network is the Feedforward Neural Network. Artificial neurons form the building blocks for these neural networks. The introduction of these neurons can be attributed to McCulloch and Pitts [1943], who developed a *perceptron* architecture to mimic the neurons in the human brain. Suppose we have an input or a set of inputs, $\mathbf{x}$, with a corresponding output y. Each individual neuron takes a weighted sum of the inputs and then passes that sum through some function, referred to as *activation function*, to produce an output, $\hat{y}$. Eq. (2.2) represents this process, where the inputs, $x_1, x_2, \ldots, x_N$, are combined with weights of the neuron, $w_1, w_2, \ldots, w_N$, and then added to an optional constant bias, b. The result is then applied to the activation function $f(\cdot)$ to produce $\hat{y}$.

$$\hat{y} = f(\mathbf{w}\mathbf{x} + b) \tag{2.2}$$

Neural networks are composed of a group of artificial *neuron*s placed alongside in layers. These layers are known as *hidden layers*, where each neuron in a hidden layer is a *hidden unit* and the output of each hidden unit is their *activation*. Layers can

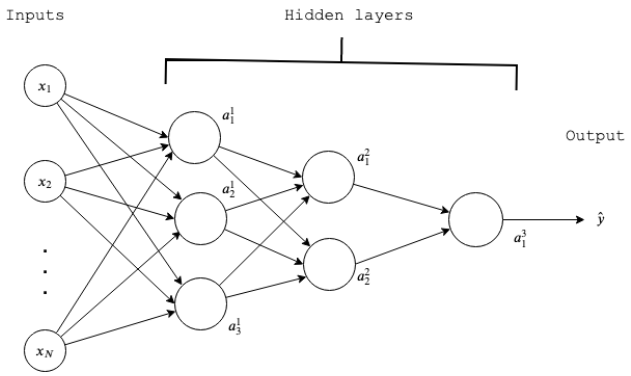

Figure 2.2: A deep neural network with two hidden layers. a_j^i represents the hidden state, or activation, of hidden unit j in the i-th hidden layer. The inputs and the estimated output are $x_1, x_2, \ldots, x_N$ and $\hat{y}$ respectively. The weights and biases have been removed from this figure for simplicity.

also be stacked to form *deep neural networks*, illustrated in Figure 2.2, that can learn complex mathematical functions. Moreover, use of *non-linear activation function*s, such as sigmoid or hyperbolic tangent, allow the neuron to learn more complex relationships, which will be more apparent as we learn about more intricate neural network models.

As we mentioned earlier, the neural networks can be *trained* in a supervised setting to learn features of labelled inputs, or training samples using their weights and biases. However, the weights and biases are generally initialised randomly and cannot produce a perfect output. In NNs, the error between the predicted output and the actual output is measured using a *loss function*, also sometimes referred to as cost function. The objective of NNs is to minimise its error, i.e. its loss function. This is achieved by updating the weights and biases of the neurons in NNs, commonly using a *backpropagation algorithm* [Rumelhart et al.].

In this thesis, we are interested in the generation and classification of sentences. As sentences are usually not the same length, simple neural networks cannot be used to process them. Moreover, these networks are not able to share features learnt across different positions of text, which is important with sentences that usually contain words or information that rely on previous words. Recurrent neural networks are examples of deep neural networks which have been attributed to learning complex relationships in sequential data such as sentences.

2.2.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) comprise of a family of neural networks that can process sequential data, such as text. These networks have cyclic, or *recurrent*, connections within layers, which means the activation at some point during the recurrency, or *state* of the RNN, depends on the activation of the previous state. A recurrent activation introduces the notion of memory in the network and also allows it to process

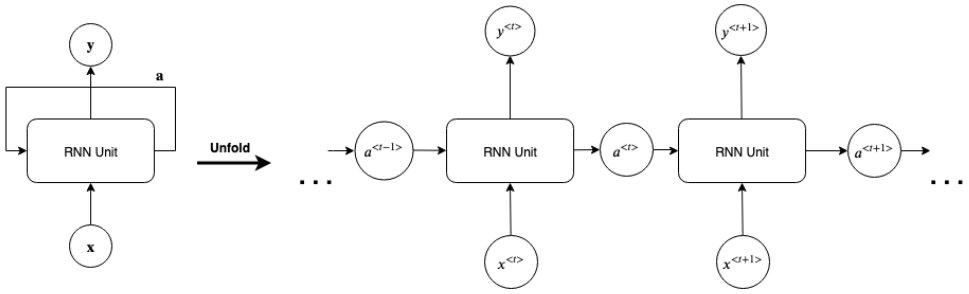

Figure 2.3: Illustration of a Recurrent Neural Network. On the left, **x** represents a sequence which can be recursively learnt using hidden state **h** to produce an output **y**. On the right, the model is unfolded to show a section of the RNN, at different time steps $T \in [\ldots, t-1, t, t+1, \ldots]$, where $x^{<T>}$, $q^{<T>}$ and $y^{<T>}$ denote the input, activation and output of a RNN unit at time step T.

a sequence of variable length. This means that the network can be modelled with time steps such that a new input can be fed into the model one step at a time to obtain corresponding outputs. Figure 2.3 illustrates this in a compact form (left) and unfolded form (right).

Although the basic RNN model creates a notion of memory with its ability to process sequential data, they perform poorly as the length of these sequences increases. This means, given long sentence(s), the model tends to forget information presented in early time steps. As conservation of *long-term dependencies* is important for modern NTG applications, (See Section 1.1) basic RNNs are less practical for tasks like language modelling, machine translation, etc.

2.2.3 Long Short-Term Memory

A Long Short-Term Memory (LSTM) network is a RNN where the recurrent RNN unit is replaced with recurrent LSTM units. These units are *gated* to maintain more persistent memory, which make it easier for RNNs to capture long-term dependencies. Using the current input, e.g. a word, each unit generates its own memory and activation based on those from the previous unit, which means the analogy of time step modelling can be applied for LSTM networks as well. The gates allow the LSTM units to "remember" important information from previous time steps and even "forget" information that may no longer be relevant. The architecture of a LSTM unit, illustrated in Figure 2.4, can be broken down into different stages. In this case, "previous" memory and activation refers to the respective values that were passed on from a LSTM unit in the previous time step. The LSTM unit then produces memory and activation values for the current time step, which is passed on to the unit in the subsequent time step.

- **Forget Gate**: By looking at the current input and the previous activation, this gate produces a value to determine the importance of remembering the previous activation.

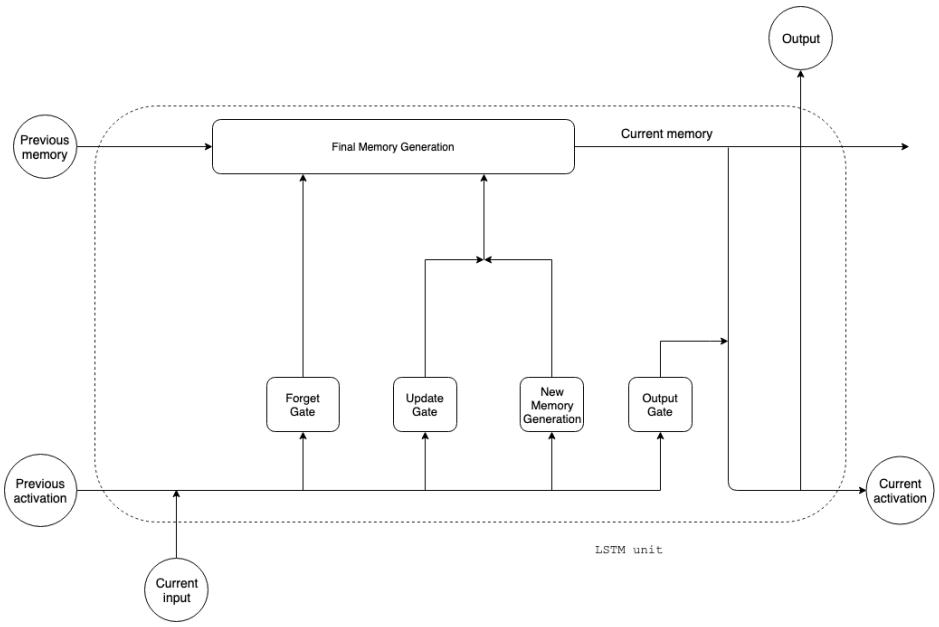

Figure 2.4: A LSTM unit. This representation was inspired by a blog post, *Understanding LSTM Networks*, by Olah [2015]

- **Input Gate**: This gate uses the previous activation and the current input to produce a measure of relevance of this input in the current time step.
- **New Memory Generation**: A new memory is generated using the current input and previous activation, which contains details about the current input.
- **Current Memory Generation**: Using the indicator from the forget gate, a decision is made to either preserve or forget the previous memory. The result is then combined with the indicator from the input gate and the new generated memory to determine the current memory for the LSTM unit. This memory is then passed on to the LSTM unit in the subsequent time step.
- **Output Gate**: This gate extracts the significant information from the current input and the previous activation. This is then combined with necessary details from the current memory to generate the current activation of the LSTM unit. The activation value can be used in two ways: (1) to produce an output for the current time step and (2) passed to the LSTM unit in the next time step.

Hence, using the memory and activation of each LSTM unit, the network learns to place importance on inputs that may have been encountered many time steps ago, which makes this architecture very useful for preserving long-term dependencies. As discussed in Section 1.1.1, LSTM networks have shown promising results in both *generative* tasks such as text-to-text generation and *discriminative* tasks like text classification. Due to it's high performance in generative tasks, the text generator used in this thesis, LeakGAN, uses LSTM networks as part of its text generation

process. The discriminative properties of LSTM are used to classify human-authored and machine-generated text (See Section 2.6.3).

2.3 Text generator architecture

In the following section, we will discuss the chosen architecture used in the generation of sentences, LeakGAN, along with reasoning behind the choice of this generator. In the context of the thesis, we do not need to know the internals of the generator since it is *loosely* a "black-box" used to generate text and can thus be replaced by any other architecture that can generate text e.g. RNN, LSTM, SeqGAN etc. Please refer to the detailed paper by Guo et al. [2017] for more information about this model.

2.3.1 Overview of LeakGAN as a text generator

At the time of this thesis, LeakGAN was the state-of-the-art GAN architecture for long text generation (See Section 2.3.2). Since our focus was to use *long synthetic sentences* to obtain useful features, to differentiate these sentences from the human-authored ones, LeakGAN was chosen as the generator in this project. The LeakGAN can be *loosely* thought of as a RNN (See Section 2.2.2) variant, where the RNN unit is replaced by a LeakGAN unit. The architecture of LeakGAN is discussed in terms of sentence generation, where the *token* at each time step could be a word, punctuation, etc. in the sentence.

The task of the generator and discriminator in a GAN text generator can be summarised as follows:

- Generator needs to generate high quality sentences to deceive the discriminator.
- Discriminator needs to identify the generated sentences as real or "fake" according to the feature extracted by itself during training.

The main motivation behind the LeakGAN architecture is the realisation that if the features used by the discriminator, are *leaked* to the generator, these would assist the generator in creating text more informatively. The model uses recent advancements in reinforcement learning (See Section 1.1.2), proposed by Vezhnevets et al. [2017] to divide the task of text generation into two parts: a Manager $\mathcal{M}$ and a Worker $\mathcal{W}$ as shown in Figure 2.5. We try to use similar notation as the authors of the architecture, with some exceptions, to facilitate comparison if required.

2.3.1.1 Discriminator $\mathcal{D}$

The discriminator consists of a *Feature Extractor*, $\mathcal{F}$, which obtains features, f_t, from the word embeddings (See Section 2.1.2) of a given sentence at each time step t. This is followed by a Deep Neural Network *Classifier* (DNNC) that can obtain a probability of the sentence being real or synthetic based on the extracted features. During the discriminator training stage, the generator $\mathcal{G}$ produces a batch of synthetic

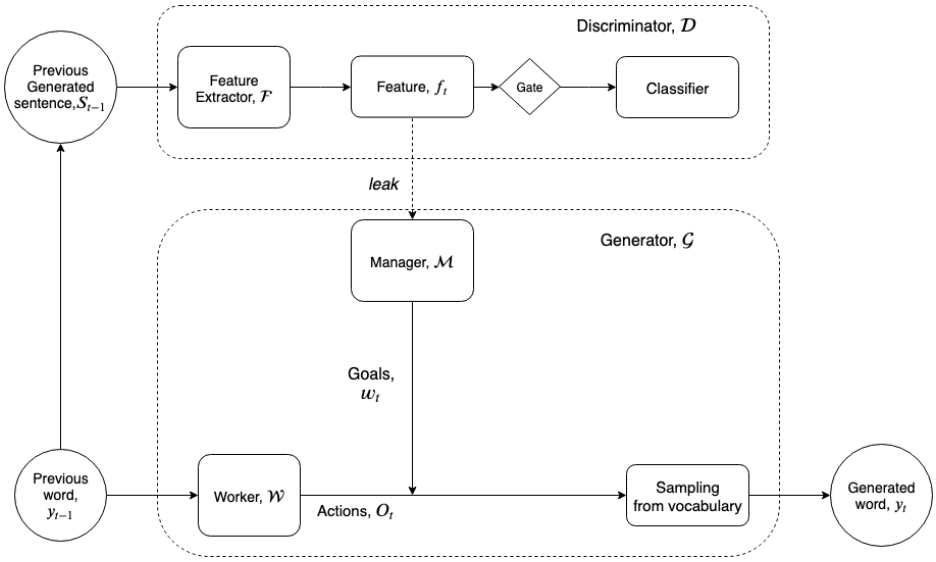

Figure 2.5: A schematic overview of LeakGAN at a time step t. The Discriminator and the Generator are in the upper and lower dotted boxes respectively. The Generator contains a *Manager* and a *Worker* module. The Manager forwards information *leaked* from the discriminator to help the worker take action, resulting in the generation of a next word by sampling from the available vocabulary.

sentences, which are then paired with a batch of labelled real sentences from the training dataset. These sentences are then used to compute the probabilities , which are the outputs of the DNNC. The weights and biases of the classifier can then be updated using backpropagation, as discussed in Section 2.2.1. Also note that the features can only be used by the DNNC once an entire sentence is generated and is, thus, ready for classification. This can be controlled by placing the *Gate* prior to the classifier which checks the status of the sentence generation.

2.3.1.2 Generator G

The Worker module uses the generated word at the previous time step, y_{t-1} to create actions, O_t. The Manager module controls the transfer of the leaked features, f_t, at each time step t. After processing the features, this module creates goals, w_t, in order to guide the output from the Worker module. These actions are then combined with the goals from the Manager module in order to choose the word to be generated at this time step, y_t, from the available words in the vocabulary. This process is labelled in Figure 2.5 as *Sampling from vocabulary*.

2.3.2 Performance of LeakGAN as a long text generator

As discussed in Chapter 1, we require a state-of-the-art text generator that can create long sentences. Guo et al. [2017] used BLEU (See Section 1.2) and Negative Log-Likelihood (NLL). NLL can be applied on synthetic text and indicates the generators'

tendency to copy real text. Lower NLL values are deemed to be better. The authors show that, in all measured metrics, LeakGAN shows significant performance gain compared to other text generation models, as presented in Table 2.1 and 2.2. These results further justify the choice of LeakGAN in this thesis for generating long sentences in comparison to existing models discussed in Section 1.1.

Text Length	MLE	SeqGAN	RankGAN	**LeakGAN**
20	9.038	8.736	8.247	**7.038**
40	10.411	10.310	9.958	**7.191**

Table 2.1: NLL performance on synthetic data.

Method	SeqGAN	RankGAN	**LeakGAN**
2-BLEU	0.8590	0.778	**0.956**
3-BLEU	0.6015	0.478	**0.819**
4-BLEU	0.4541	0.411	**0.627**
5-BLEU	0.4498	0.463	**0.498**

Table 2.2: n-BLEU scores on test data of EMNLP2017 WMT

2.4 Text Evaluation Metrics

In this section, we discuss the metrics used in this thesis to quantitatively evaluate real and synthetic sentences. In this thesis, we placed higher priority in metrics that can potentially assist in classification of human-authored and machine-generated text.

2.4.1 Novelty

Novelty can be used to investigate the dissimilarity of a sentence in a set of sentences, S_1, with the sentences in another set, S_2 , it is a **between-set** metric. In other words, novelty of generated sentences can help figure out if the generator simply copies the sentence in the corpus instead of generating new ones if we calculate the novelty of generated sentences against sentences in training corpus.

In this thesis, a novelty score between 0 and 1 inclusive indicates the distance of each sentence from the most similar sentence in the other set. For instance, high novelty of a generated sentence implies that the sentence is very different from the most similar sentence in the set of training (real) sentences and vice versa.

The novelty of a sentence S_i, given another set of sentences C is

$$Novelty(S_i) = min\{\delta(S_i, C_j)\}_{j=1}^{j=|C|} \tag{2.3}$$

where δ is a function that calculates distance (or dissimilarity) between two sentences.

2.4.2 Diversity

Diversity can be used to investigate the dissimilarity of a sentence in a set of sentences with the rest of sentences in the same set, it is a **within-set** metric. Therefore, when we calculate the diversity generated sentences, it's value can indicate if the generator can produce a variety of sentences.

In this thesis, a diversity score between 0 and 1 inclusive indicates the distance or dissimilarity of each sentence from the most similar sentence in the same set. Therefore, high diversity of a generated sentence implies that the sentence is very different from the most similar sentence in the set of generated sentences and vice versa.

Given a collection of sentences S, the diversity of sentence S_i can be defined as

$$Diversity(S_i) = min\{\delta(S_i, S_j)\}_{j=1}^{j=|S|, j\neq i} \tag{2.4}$$

where δ is a function that calculates dissimilarity between two sentences.

2.5 Text Dissimilarity Functions

This section discusses the various functions used to calculate dissimilarity between two text bodies, sentences in this case. These functions, applied to words, has been used in evaluation metrics namely *Novelty* in Section 2.4.1 and *Diversity* in Section 2.4.2.

2.5.1 Jaccard Distance

Jaccard index is a metric often used for comparing similarity, dissimilarity, and distance between data sets [Niwattanakul et al.]. In this thesis, we use Jaccard distance, which is defined as a dissimilarity measurement between data sets. Jaccard distance between two data sets, S_1 and S_2, is the result of division between the number of uncommon words between the two sets (symmetric difference, Δ) divided by the number unique words in both sets.

$$\delta(S_1, S_2) = \frac{|S_1 \Delta S_2|}{|S_1 \cup S_2|} \tag{2.5}$$

In a rare case, if both the data sets are empty, the similarity coefficient is defined to be 1, which implies that

$$0 \leq \delta(S_1, S_2) \leq 1$$

A high Jaccard distance of generated sentence when compared to a human-written sentence provides a measure of how much the text generator has copied from that sentence. The *time complexity* of our implementation is $O(|S_1| \times |S_2|)$, where S_i represents the number of words in a set S_i.

2.5.2 Normalised Levenshtein Distance

Given two sentences S_1 and S_2, the Levenshtein distance [Levenshtein, 1965] between them is the minimum number of *edit operations* needed to transform S_1 into S_2. The allowed edit operations in this case are:

1. add a word to a sentence,
2. remove a word from a sentence, and
3. substitute a word in a sentence by another word.

Since this metric does not obey the range [0,1] we use a normalised version of Levenshtein distance (NLD) in this thesis

$$NLD(S_1, S_2) = \frac{Levenshtein(S_1, S_2)}{max(|S_1|, |S_2|)} \tag{2.6}$$

where $|S_i|$ represents the number of words in a sentence S_i. The *time complexity* of our implementation is $O(|S_1| \times |S_2|)$. NLD between two sentences provides a

measure of number of common words, the order of these words and the difference in length between these sentences.

2.5.3 Word Mover's Distance

As discussed in Section 2.1.2, *Word embeddings* are representations of words as low-dimensional vectors in which similar words are expected to be close in the vector space [Collobert and Weston, 2008]. Words in this representations are referred to as *embedded words*. The *Word Mover's Distance* (WMD) [Kusner et al., 2015] measures the dissimilarity between two text documents, in this case sentences. The authors of the metric describe it as "the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document". The metric uses the *word2vec* architecture [Mikolov et al., 2013a] that generates word embeddings. The model learns a vector representation for each word using a NN architecture. Each of these vectors is trained to maximise the probability of neighbouring words in a corpus. The authors of WMD also demonstrate that distances between embedded word vectors are semantically meaningful *e.g.* vec(*Berlin*) - vec(*Germany*) + vec(*France*) is close to vec(*Paris*).

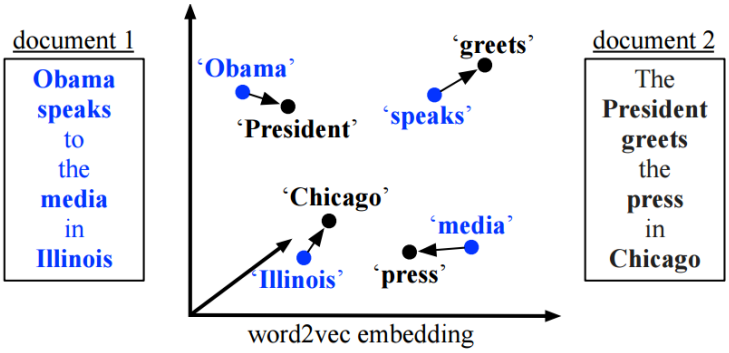

Figure 2.6: Visual representation of *word mover's distance* [Kusner et al., 2015]. The *document*s in this diagram are referred to as sentences throughout the thesis.

Given a word2vec embedded matrix $\mathbf{X} \in R^{d \times n}$, where d is the dimension of the embedding, and n is the number of words in the entire vocabulary. The i-th column, $x_i \in R^d$, represents the embedding of the i-th word.

Each sentence is represented as a normalised *Bag-Of-Words* (BOW)[1] vector $\mathbf{d} \in R^n$, and $\mathbf{d}_i = \frac{c_i}{\sum_i c_i}$, where i's denote the word tokens and c_i is the number of times i appears in the document. The distance between words are the *Euclidean distance* of their embedded word vectors, denoted by $c(i, j) = ||\mathbf{x}_i - \mathbf{x}_j||_2$, where i and j denote word tokens.

The distance between sentences, WMD, is defined as the minimum weighted cumulative cost needed to move all the words from $\mathbf{d}$ to $\mathbf{d}'$. by $\sum_{i,j} \mathbf{T}_{ij} c(i, j)$, where

[1]The Bag-Of-Words approach is a very common feature extraction procedure for sentences and documents, which looks at the histogram of the words within the text, i.e. each word count is considered as a feature. [Manning et al., 2008]

$\mathbf{T} \in R^{n \times n}$. Each element $\mathbf{T}_{ij} \geq 0$ denotes how much of word i in the first sentence, $\mathbf{d}$, travels to word j in $\mathbf{d}'$.

The WMD solution is provided by the following minimisation problem:

$$min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j) \tag{2.7}$$

given the constraints: $\sum_{j=1}^{n} \mathbf{T}_{ij} = d_i$, and $\sum_{i=1}^{n} \mathbf{T}_{ij} = d'_j$.

The WMD has several useful properties [Kusner et al., 2015]:

- It does not have hyperparameters, which makes it relatively easy to implement, as no tuning is required for model;
- It can be interpreted as the distance between individual words which sum up to determine the distance between sentences containing these words;
- It automatically integrates the information encoded in the *word2vec* space, which leads to high retrieval accuracy.

In comparison to Jaccard distance and Normalised Levenshtein distance, this metric promises to inject semantic meaningfulness towards classification of real and machine-generated text as it leverages *word2vec* model as discussed above. However, the *time complexity* of this function is $O(p^3 \log p)$ where p is number of unique words in the sentences to be compared.

2.6 Classifiers used in the Classification of Sentences

Classification is a form of data analysis that constructs models, known as *classifiers* or *classification model*s, to describe data classes. Data classification consists of two steps: a learning step where a classification method is constructed from a labelled **training set** and a classification step where the model is used to predict class labels for a given data, **test set**. In this thesis, the classifiers are designed to predict if a given sentence or its features belong to one of two classes: real or synthetic.

In this section, we will discuss the three different classifiers used in this thesis: Naïve Bayes, Random Forest and LSTM classifier. In the following section, we only discuss the classification details that are important in this thesis. If the reader that wishes to learn more about classification techniques, please refer to the book *Data Mining Concepts and Techniques* by Han et al. [2011].

We will use the following notation in the following subsections. The training set consists of samples and their corresponding class labels. A sample, $\mathbf{X}$, can be represented by a n-dimensional feature vector, $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ depicting n features, where each sample is labelled as being in the real or in the synthetic class.

2.6.1 Naïve Bayes classifier

In this thesis, a Naïve Bayes classifier is *trained* on novelty and diversity of labelled sentences for each of the text dissimilarity functions: Jaccard, Normalised Levenshtein and Word Mover's Distance. During *testing*, the classifier is asked to predict if a sentence is either real or synthetic given its novelty, diversity or both. This is a "glass-box" classifier, which means predictions made with this model can be interpreted by observing the distribution of values in the dataset.

Bayes (or Bayesian) classifier, an instance of statistical classifiers, can predict class membership probabilities such as the probability that a given sample belongs to a certain class. Before we discuss, we need to briefly review Bayes' theorem. We will assume that the reader is familiar with basic probability theory (e.g. conditional probability), including notations and statistical measures such as mean and standard deviation.

Suppose we have a hypothesis H that a sample $\mathbf{X}$ belongs to a class C. Bayes' Theorem states that

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} \tag{2.8}$$

where,

- $P(H|\mathbf{X})$, *posterior probability*, is the probability that the hypothesis H holds given the sample $\mathbf{X}$
- $P(\mathbf{X}|H)$ is the posterior probability of $\mathbf{X}$ conditioned on H
- $P(H)$, *prior probability*, is the probability of H regardless of $\mathbf{X}$
- $P(\mathbf{X})$ is the prior probability of $\mathbf{X}$

The Naïve Bayes classifier can now be constructed as follows. If there are m classes, $C_1, C_2, \ldots, C_m$, the classifier will predict that a sample $\mathbf{X}$ belongs to the class having the highest posterior probability, conditioned on $\mathbf{X}$.

$$P(C_i|\mathbf{X}) > P(C_j|\mathbf{X}) \text{ for } 1 \leq j \leq m, i \neq j \tag{2.9}$$

Therefore, we maximise $P(C_i|\mathbf{X})$. By Bayes' theorem,

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})} \tag{2.10}$$

Since $P(\mathbf{X})$ is constant for all classes, we only need to maximise $P(C_i|\mathbf{X})$. Moreover, Naïve Bayes Classifier assumes **class-conditional independence**, which presumes that feature's values are conditionally independent of one another, given the class label of the sample. Thus,

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{2.11}$$

We refer to x_k as the value of the feature k for sample $\mathbf{X}$. In this thesis, we are dealing with continuous valued attributes for novelty and diversity for each text distance functions, so the classifier needs to assume a Gaussian distribution μ and standard deviation σ defined as:

$$G(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.12}$$

We can then compute $P(\mathbf{X}|C_i)$ using Eq. (2.11) by calculating each $P(x_k|C_i)$ as

$$P(x_k|C_i) = G(x_k, \mu_{C_i}, \sigma_{C_i}) \tag{2.13}$$

where μ_{C_i} and σ_{C_i} are the mean and standard deviation, respectively, of the values of each attribute for training samples of class C_i.

Finally, in order to predict the class label of $\mathbf{X}$, $P(\mathbf{X}|C_i)P(C_i)$ is calculated for each class C_i. The classifier then predicts that the class label of sample $\mathbf{X}$ is the class C_i if and only if,

$$P(\mathbf{X}|C_i)P(C_i) > P(\mathbf{X}|C_j)P(C_j) \text{ for } 1 \leq j \leq m, i \neq j \tag{2.14}$$

Han et al. [2011] state that Naïve Bayes classifier is *comparable* in performance with classifiers such as decision trees (discussed in the following section) and selected neural network classifiers. The simplicity of this model as well as the ease of interpretability, using probability distribution of continuous classification features, make it a practical classifier to be used in this thesis.

2.6.2 Random Forest classifier

A Random Forest classifier is more complex compared to Naïve Bayes. It is a "black-box" classifier in the sense that predictions by this model cannot be interpreted easily without further investigation. The training and test set for this classifier are the same as those for Naïve Bayes.

Random Forest classifiers [Breiman, 2001] belong to a class of classification techniques known as *ensemble methods*, where an *ensemble* refers to a combination of multiple simple, or base, classifiers, that aim to generate an advanced composite classification model [Han et al., 2011]. Given a data set, different training sets are created, where each set is used to retrieve individual classifiers. When a new sample is given to the ensemble to classify, its class prediction is based on the individual class predictions from the base classifiers. According to Han et al. [2011], an ensemble is more accurate than its base classifiers, as it will only misclassify a given sample only if more than half of its base classifiers have misclassified the sample. Random forest is an ensemble of *decision tree classifiers* and is generated using a *random* selection of features at each step.

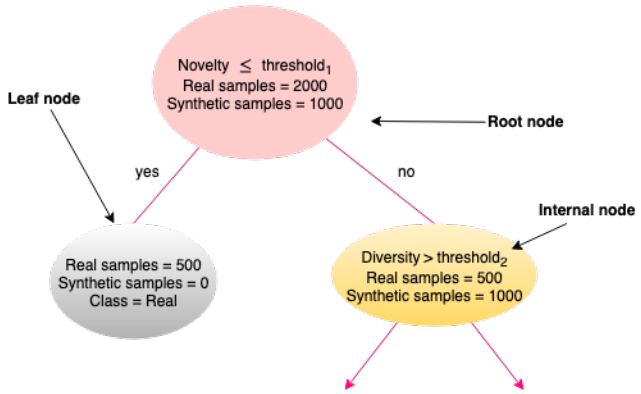

Figure 2.8: An example of a simplified hypothetical decision tree applied to the classification of *Real* and *Synthetic* sentences. The classifications features are *Novelty* and *Diversity*, which are split on $threshold_1$ and $threshold_2$ respectively.

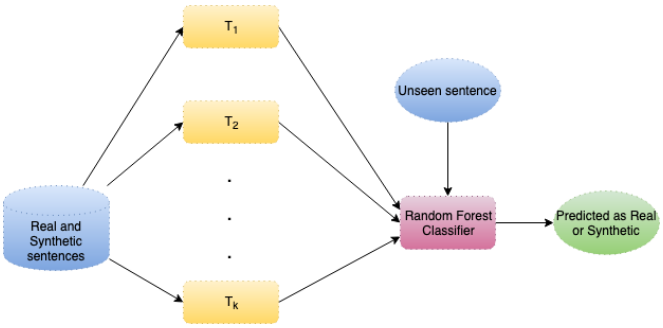

Figure 2.7: An example of a Random Forest classifier applied to the classification of *Real* and *Synthetic* sentences. The decision tree classifiers $T_1, T_2, \ldots, T_k$ are trained using *Real* and *Synthetic* sentences. When given an unseen sentence, the Random Forest classifier combines the votes from each classifier and returns a class prediction.

Figure 2.8 shows an example of a hypothetical decision tree classifier applied to the classification of real and synthetic sentences. In general terms, a decision tree is a tree where each internal *node*, or non-leaf node, denotes a test on a classification *feature*, each branch represents a *decision*, the outcome of the test, and each leaf node holds an class prediction. During training, the tree is generated in a top-down recursive manner. At the beginning, all the training samples are at the root node. The tree then looks for features and thresholds that best splits the samples into the different labelled classes. The process is repeated recursively until data has been split into uniform (or mostly uniform) groups. At that stage, the decision tree predicts the class with the majority of data in that leaf node. Given a new sample to classify, the tree traces its branches with the feature values of the sample, which were based on the thresholds created during training, leading to a class prediction.

In this thesis, the individual decision trees were produced using a random selection of features at each node to decide the split at the training stage. During

classification, as mentioned with ensemble methods, each of the decision trees vote and the Random Forest classifier makes a prediction based on the most votes, as illustrated in Figure 2.7.

2.6.3 Long-Short Term Memory Classifier

An LSTM classifier is also used in this thesis to differentiate real and synthetic sentences.In contrast to Naïve Bayes and Random Forest classifiers, we use the labelled sentences themselves as the training set, where the model uses the words in the sentences as features. The model is then asked to predict if a sentence, not seen in the training set, is either real or synthetic.

Section 2.2.3 discusses the general structure of an LSTM network. The previous architecture is known as a variant of *many-to-many* sequence prediction [Karpathy, 2015], where the network produces an output for each input word to the network. During classification, we want a single output, a class label, for the entire sentence. Karpathy [2015] refers to such an architecture as *many-to-one*. We will be using this many-to-one architecture during our implementation of LSTM classifier, illustrated in Figure 2.9.

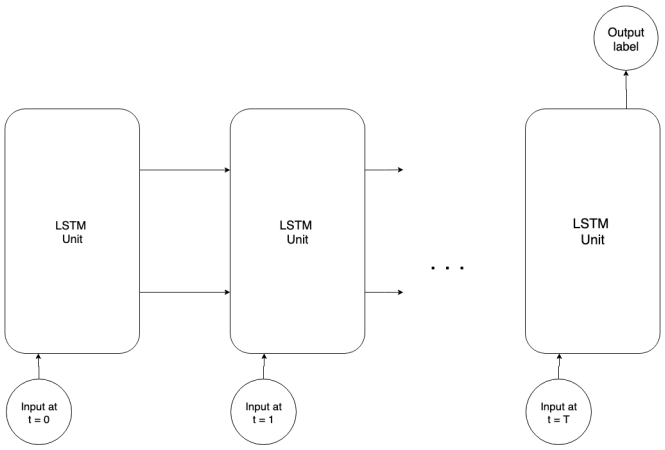

Figure 2.9: A *many-to-one* LSTM network architecture. For the classification of a sentence, each word will fed as input to an LSTM unit at each time step, t, until time step T when the final word is processed by the network. The output label for the entire sentence will be produced at that point.

The main motivation in using this classifier is to explore presence of long-term dependencies in text and whether these can be helpful in differentiating real and synthetic sentences. A high performance of this classifier could provide incentive towards future research into discovering features from these sentences. The details of an implementation of this classifier will be described in Section 3.7.3.

2.7 Evaluation of Classifier performance

In this section, we will discuss metrics used in this thesis to determine performance of classifiers in the classification of human-written and machine-generated text.

2.7.1 Stratified Cross-validation

When implementing classification models, it is important to avoid training and testing on the same dataset to avoid *overfitting*. This is a situation when a model simply repeats the labels of the training samples to achieve misleading overoptimistic scores but fails to predict unseen data. In order to avoid overfitting with Naïve Bayes and Random Forest classifiers, we implemented *stratified k-fold cross-validation*, with a recommended split of k = 10 [Han et al., 2011]. In a 10-fold cross-validation, the initial dataset, D, is randomly divided into 10 mutually exclusive groups or folds, $D_1, D_2, \ldots, D_{10}$. The classifier is trained and tested 10 times, where each iteration i uses a fixed D_i as the test set and the rest of the folds are used to train the model. In stratified cross-validation, the class distribution of the samples in each fold is kept approximately equal to that in the initial data.

2.7.2 Metrics for evaluation of classifier performance

In the subsections, for each of the classes, we will use the term **positive samples** (P) as the samples of the main class of interest (C) and **negative samples** (N) as the remaining samples.

- **True positives** (TP): positive samples that were correctly labelled by the classifier
- **False Negatives** (FN): positive samples that were incorrectly labelled by the classifier
- **True Negatives** (TN): negative samples that were correctly labelled by the classifier
- **False Positives** (FP): negative samples that were incorrectly labelled by the classifier

2.7.2.1 F1-score

We can simply calculate the **accuracy** of a model by calculating the percentage of the total samples labelled correctly by the classifier.

$$accuracy = \frac{TP + TN}{P + N} \tag{2.15}$$

However, in cases where there exists a **class imbalance problem**, where values in one class may be significantly less frequent than the other(s), we use the following metrics:

- **Recall**: percentage of positive samples that are labelled correctly. A high value of recall, 1.0, indicates that every sample from the main class of interest, C, were labelled correctly by the classifier.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{2.16}$$

- **Precision**: calculates the percentage of samples labelled as positive that are indeed positive. A high value of precision, 1.0, indicates that every sample that were labelled by the classifier as belonging to the class C in fact belongs to that class.

$$precision = \frac{TP}{TP + FP} \tag{2.17}$$

In practice, precision and recall are often combined into a single measure named F1-score, which gives equal weight to both precision and recall. Precision and recall tend to have an inverse relationship, where it is possible to increase one at the cost of decreasing the other, which makes F1-score a better alternative since it does not increase if precision or recall is improved at the expense of the other.

$$F\text{-}score = \frac{2 \times precision \times recall}{precision + recall} \tag{2.18}$$

A high F1-score of 1.0 indicates that the classification models attains both high precision and recall.

2.7.2.2 Area Under the Curve (AUC) for Receiver Operating Characteristic (ROC) curves

Given a test set and a classification model, true positive rate (TPR) is the proportion of positive samples that are correctly labelled by the classifier ($TPR = \frac{TP}{P}$) and false positive rate (FPR) is the proportion of negative samples that are mislabelled as positive ($FPR = \frac{FP}{N}$). Since ROC curve is only used for model comparison in this thesis, the details of how to analyze a ROC curve is explored below, without discussing how a ROC curve is plotted, which is not necessary for the purpose of analysis.

In this thesis, we have a two-class problem, the ROC curve allows us to visualise the trade-off between the rate at which the model can accurately identify positive samples against the rate at which it mistakenly assigns positive labels to negative samples for different portions of the set. Any increase in TPR occurs at the cost of an increase in FPR. The area under the curve (AUC) of the ROC curve is a measure of the model accuracy.

Figure 2.10: ROC curves for two sample classification models, M_1 and M_2. The diagonal shows where, a model may be equally likely to label a sample as either positive or negative, which is as good as guessing.

In this thesis, we compare two classification models using ROC curves and their associated AUC score. A classification model with a AUC score of 1.0 indicates perfect accuracy, while the closer the score is to 0.5, the less accurate the corresponding model is. Figure 2.10 shows a visual representation of how two sample classification models can be compared in terms of accuracy. The closer a ROC curve is to the "Random guessing" line, the less accurate the model is. In this case, M_1 can be seen as more accurate supplemented by a higher AUC score.

2.8 Summary

This chapter has provided the theory required to explain the models that will be used in the generation of synthetic sentences. It also introduces the reader to the architectures, which will be built to classify human-authored and synthetic sentences, trained with text evaluation metrics as their features. Finally, we have discussed metrics that will be used to examine performance of the classification models. The next chapter outlines the complete experimental setup used in this thesis. Details regarding the dataset used, corresponding preprocessing as well as the implemented methods for evaluation of the setup are described appropriately.

Chapter 3

Experiment Setup and Methodology

This chapter provides details on the experimental setup to generate long synthetic sentences using LeakGAN. Steps taken during calculation of the text evaluation metrics, i.e. novelty and diversity, for both human-authored and synthetic sentence are also provided. Details regarding the construction of different classifiers to differentiate real and synthetic sentences are also presented. These classifiers the text evaluation metrics and sentences as features.

Section 3.1 provides a discussion of the initial experiment setup justified by the objectives of this thesis.

Section 3.2 provides an analysis of the dataset chosen for generation of text with justification behind this choice.

Section 3.3 describes the steps taken to prepare the dataset for training.

Section 3.4 outlines the steps taken to split the processed dataset into two distinct datasets: (1) to train the generator, and (2) to be used during feature extraction of real sentences.

Section 3.5 provides a brief analysis of the generated text and subsequent filtering methods to extract necessary sentences.

Section 3.6 discusses the text processing and any methodology employed during the calculation of text evaluation metrics for each of the text dissimilarity functions: Jaccard distance, Normalised Levenshtein distance (NLD) and Word Mover's Distance (WMD).

Section 3.7 outlines the steps taken in training and testing classification models to differentiate real and synthetic text.

3.1 Initial Experimental setup

As mentioned in Chapter 1, we are interested in extracting features from both human-authored and machine-generated sentences that can be used to classify these two forms of text. We aim to achieve this by comparing human-authored, or real, sentences and machine-generated, or synthetic, sentences. These features can then be used as parameters to train different classifiers and investigate their adequacy in differentiating real and synthetic sentences by examining the performance of these classifiers.

As a consequence, we need both real sentences and synthetic sentences which can then be evaluated (See Section 2.4) using text dissimilarity functions (See Section 2.5). Real sentences can be extracted from an available text corpus, while synthetic sentences need to be generated using a text generator, which is preferably state-of-the-art so as to obtain high quality sentences. Moreover, we want to focus on long sentences, with at least 20 words, which means both real and synthetic sentences need to be processed in order to maintain this specific sentence length throughout the experiment.

An initial setup of the experiment, based on the requirements outlined above, can be visualised in Figure 3.1. The black boxes represent parts of the experimental setup that will be unveiled in the following sections in this chapter.

Figure 3.1: Initial setup for experiment showing the original dataset and the classified real and synthetic sentences as the final product.

3.2 The Dataset

In the field of text generation, it is common to use a few *selected* datasets for model training and evaluation in order to maintain consistency during comparison of model performance. As discussed in Chapter 2, we will be using the LeakGAN as the GAN-based text generation model. The LeakGAN authors have used COCO Image Captions Dataset [Chen et al., 2015] and the EMNLP2017 WMT4 Dataset [Bojar et al., 2017], which are then used for comparison with other GAN-based text generation models. Amongst the two datasets, the authors have exclusively used the News section from the EMNLP2017 WMT4 Dataset for the generation of long text. As we are focused on only long sentences in this thesis, we have chosen this particular dataset as our *Real Text Corpus* (See Figure 3.1). The news dataset consists of 646,459

We ′ ve only been recording them for the last month or so , but now that word has spread we ′ re getting daily reports - people finding eight or 10 at a time .
The pair were taken to the San Francisco police ′ s Park Station and will eventually be moved to the city jail .

Table 3.1: A sample of sentences from EMNLP2017 WMT4 Dataset as preprocessed by the authors of this dataset

words and 397,726 sentences. The sentences in this dataset were processed by the authors of the dataset to remove news headlines, author names and other sequences that may not form proper sentences structures. They also introduced a whitespace between each character and a punctuation, which meant that *contraction*s[1] such as "don't", "I've" were transformed to "don ′ t", "I ′ ve" respectively. Examples of such sentences are shown in Table 3.1 and more samples are available in Table A.1.

3.3 Preprocessing and Filtering of the Dataset

The procedure for training the LeakGAN, in this thesis, was designed to mimic the training guidelines used in the original paper by Guo et al. [2017]. The generator requires a set of tokenised sentences to train the model. As the sentences in the original dataset were not tokenised in that form, some preprocessing was required in order to conform to a data structure suitable to be used for training the LeakGAN.

3.3.1 Extraction and tokenisation of sentences

The first step in the preprocessing was extracting sentences from the dataset. We implemented this using the *Natural Language Toolkit* (NLTK) [Loper and Bird, 2002], which is a widely used Python toolkit for various text processing tasks.

The next step in the preprocessing was tokenisation of sentences into words, introduced in Section 2.1.1. A simple approach of splitting a sentence on spaces and newlines is quite limited as it does not handle capital words well. Suppose we have a sequence as given in the pre-processed dataset: *You ′ re going there , aren ′ t you ?*. In the first token of the sequence, "You", the capital "Y" means that the token is not equal to "you", which may cause issues when the token is used for generator training and calculation of text dissimilarity functions. This is due to errors introduced in counting words, which influences Jaccard distance (See Section 2.5.1), and during insertion, deletion or replacement of words, which affects Levenshtein distance (See Section 2.5.2). To overcome these limitations, we applied lower-casing to all words in the sequence before splitting into tokens. This resolved the issue of "You" and "you" being different tokens. Table 3.2 demonstrates the results of the tokenisation and lower-casing on the aforementioned sentence: *You ′ re going there , aren ′ t you ?*.

[1] A contraction is a word constructed by shortening and combining two seperate words.

Tokenisation Technique	Sentence/Tokenised Sentence
No tokenisation	You ’ re going there , aren’ t ’ you ?
Simple tokenisation	[’You’, "’", "re", ’going’, ’there’, ’,’, ’aren’, "’", ’t’, ’you’, ’?’]
Simple tokenisation with lower-casing	[’you’, "’", "re", ’going’, ’there’, ’,’, ’aren’, "’", ’t’, ’you’, ’?’]

Table 3.2: An example of a sample sentence in the preprocessed dataset before and after tokenisation

Figure 3.2: Experimental setup for preprocessing and filtering of the original dataset

It is worth mentioning that the choice of using lower case for all tokens introduces an issue that are now treated as the same. One example is a name like "Trump" which, when lower cased to "trump", mean something different. Lower-casing was still preferred in this thesis as there were more cases in the given dataset where lower-casing was required to reduce ambiguity.

3.3.2 Filtering the dataset

Following the preprocessing procedure of LeakGAN by Guo et al. [2017], the sentences were then filtered by eliminating words appearing less than 4,050 times along with all sentences containing these words. In order to focus on long sentences, sentences containing less than 20 words were also removed. After the filtering, the news dataset had 5,705 words and 304,222 sentences with a maximum sentence length of 51. Figure 3.3 shows the distribution of sentence length in the original dataset before and after preprocessing and filtering steps outlined above. The experimental setup at this stage is illustrated in Figure 3.2.

A very interesting aspect of this dataset were the frequently referenced topics. Since this is a news dataset from the year 2017, the major topics revolved around "Brexit", "Hillary Clinton", "President Barrack Obama", "Donald Trump", "U.S. elec-

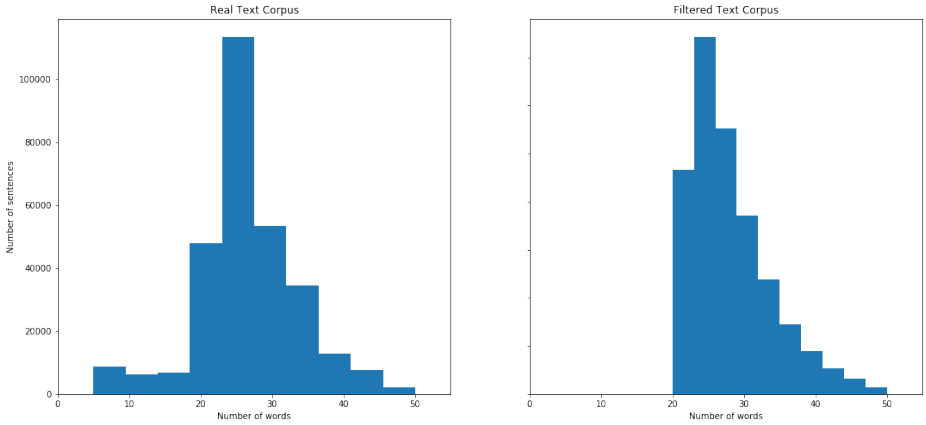

Figure 3.3: Distribution of sentence length after preprocessing and filtering *Real text corpus* to remove sentences less than 20 tokens.

tions", etc. The dominance of these topics became more evident in this section since elimination of less frequent words and sentences containing these frequent words failed to filter these topics.

3.4 Splitting the processed dataset for training and classification

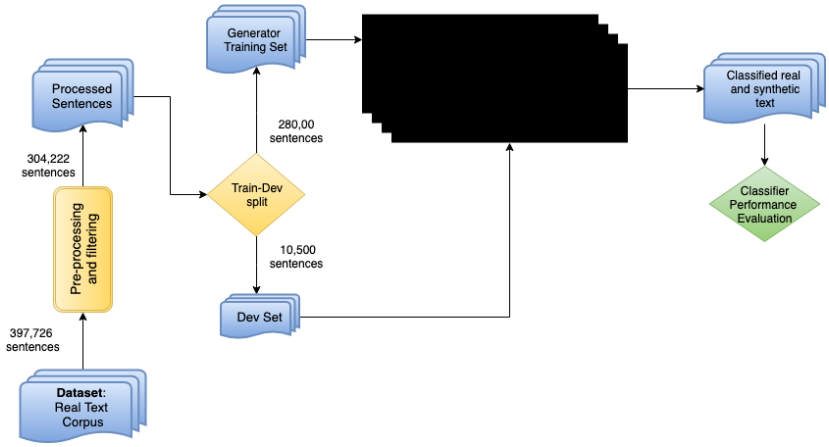

Figure 3.4: Experimental setup for splitting the pre-processed and filtered sentences using *Train-Dev Split* into *Generator Training Set*, to be used for training the text generator and *Dev Set*, which is set aside to be used during classification of real and synthetic sentences.

After preprocessing and filtering the dataset, the tokenised sentences were now available in a format that could be used by the LeakGAN to generate sentences. As mentioned in Section 3.1, we will eventually need to compare real sentences with

synthetic sentences in order to calculate evaluation metrics for classifying the two. Since we do not want the generation of these synthetic sentences to be influenced by the real sentences they will be compared to, we need to maintain a separate set of randomly selected real sentences that can be used for comparison. This process is illustrated in Figure 3.4 as *Train-Dev Split*, where we randomly sample 280,000 sentences as the *Generator Training Set* and another 10,500 sentences as the *Dev Set*. The Dev Set is set aside to be used during the training and testing stage of the classification model, further discussed in Section 3.7.

We chose 280,000 sentences as number of training samples in the Generator Training Set in order to leave the remaining samples for testing. However, the number of samples in the Dev Set was not chosen until later in the experiment when we were calculating dissimilarity metrics for sentences. The high complexity of text dissimilarity functions, especially that of Word Mover's Distance discussed in Section 2.5.3 as well as Normalised Levenshtein distance (See Section 2.5.2), made it difficult to increase the sample size beyond 10,500 during the evaluation of real and synthetic sentences.

3.5 Generation and filtering of synthetic text

After obtaining the *Generator Training Set*, the LeakGAN was trained on this dataset in default settings set by its authors, using the code available on Github, to generate a set of 15,000 sentences. This set of synthetic sentences is referred to as *Synthetic Set*. Please note that, following the training settings by Guo et al. [2017], the LeakGAN was trained with CUDA 7.5.17 for GPU optimisation. With this setting, the training and generation of synthetic text still took approximately two weeks.

Table 3.3 shows a small sample of sentences generated by the LeakGAN. In this small sample, the sentences do not seem to convey perfect sense, although grammatical errors tend to be minimal. More examples of synthetic text are included in Table A.2. The *Synthetic Set* initially consisted of 4,764 words and 15,000 sentences. These words are all in lower case since the LeakGAN was trained on lower case words only. As we are only concerned with long sentences in this thesis, any short sentences, containing less than 20 words were now filtered out. After the filtering process, the *Synthetic Set* has 4,082 words and 14,055 sentences with a maximum length of 49. As mentioned earlier, we were constrained by the time complexity of text dissimilarity functions, especially Word Mover's Distance. Hence, we chose 10,500 sentences from the filtered *Synthetic Set* to evaluate their novelty and diversity. A distribution of sentence length prior to and after this filtering is illustrated in Figure 3.6.

The sentences in *Synthetic Set* and in the *Dev Set* are now labelled respectively, as illustrated in Table 3.4 and are used to calculate corresponding evaluation metrics as discussed in Section 3.1.

you need to have a desire to find that i can do it , ” he said .
there ’ s always a yellow advance and we have to miss the way , it ’ s going to be a good time .
the state department has acted , but because he would be from the most powerful students who says illegal that is not true .
the deputy prime minister was more likely to be treated as a terrorist attack , he would meet with the taliban .
the survey said that the message is a concern but that they have to be the same actor , which can be a result .
it has been really a spokesperson and no top thing are reviewed that practice or how i think russell is too high or we deserve one time .

Table 3.3: Sample of synthetic text after training LeakGAN

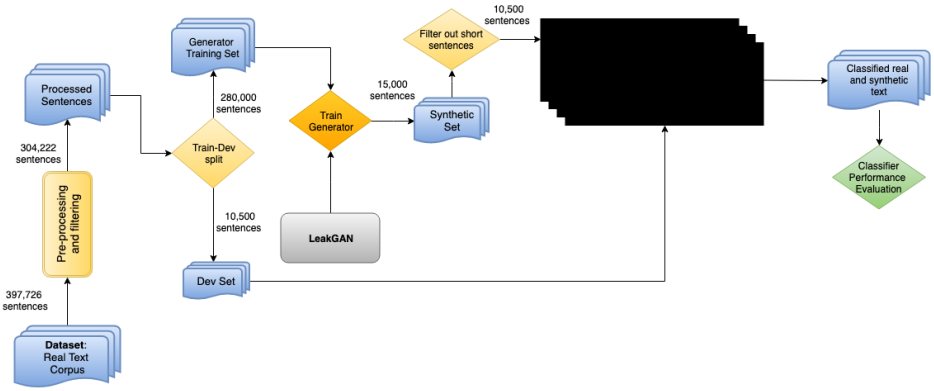

Figure 3.5: Experimental setup used during the generation of synthetic sentences, *Synthetic Set*, using the LeakGAN by training on *Generator Training Set*. The synthetic sentences, filtered to remove shorter sentences with less than 20 words, can then used during classification of real and synthetic sentences.

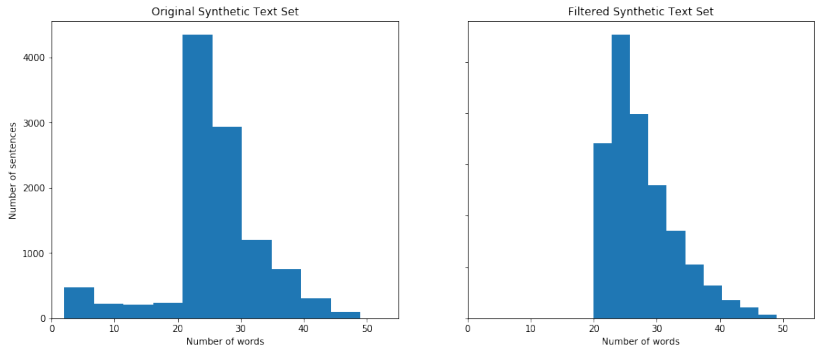

Figure 3.6: Histogram to represent the frequency of sentences against number of words per sentence for *Synthetic Set* before and after filtering to remove sentences with less than 20 words.

text	label
"the scottish government has won and not necessarily only the federal reserve fund , and a real estate deficit is the best way ."	0
"we need to be , but " that ' s just a chance to change the value of the united states and is likely to be better ."	0
"we saw the label , but in life , and the source said , they know that they did not leave the eu unless the government has not yet to give them a " financial possible opportunity to pay and the biggest person in the world"	0
"they have to send the form back , they do not get the option of 25 meetings with 17 ministers to decide what their rate of tax is ."	1
"the force ' s civilian watchdog was also warned that the police and partner authorities had yet to draw up a consistent list of vulnerable sites , according to the private paper , seen by the press association ."	1
"but they ' re going to monitor her developing baby to see if the baby has been affected and once that baby is born they ' ll do hearing tests , vision tests to see if the baby was damaged ."	1

Table 3.4: A sample of labelled sentences. A *label* value of 0 indicates the sample belongs to the *Synthetic Set*, while a value of 1 indicates the sample belongs to the *Dev Set*.

3.6 Calculation of Text Evaluation Metrics

The labelled Dev Set and the Synthetic Set were used to calculate the text evaluation metrics, *Novelty* and *Diversity* (See Section 2.4), for each sentence in both sets. The following subsections discuss the text processing and any other methodology employed during the calculation of these metrics for each distance function: Jaccard distance, Normalised Levenshtein distance and Word Mover's Distance (See Section 2.5). Note that the words in both the datasets are already in lower case.

3.6.1 Jaccard distance

Since this distance function is influenced by the number of common words during the comparison of sentence dissimilarity, punctuations are removed from both the datasets. No further processing or transformation (e.g. lemmatization[2]) is applied in order to preserve the originality of words produced by the generator.

3.6.2 Normalised Levenshtein distance

Similar to Jaccard distance calculation, only punctuations are removed from both the datasets. Normalised Levenshtein distance is affected by the order as well as the number of words in a sentence, which means common methods of preprocessing

[2]*Lemmatization* aims to remove inflectional endings only and to return the base or dictionary form of a word, known as the *lemma*, with the use of a vocabulary and morphological analysis of words.

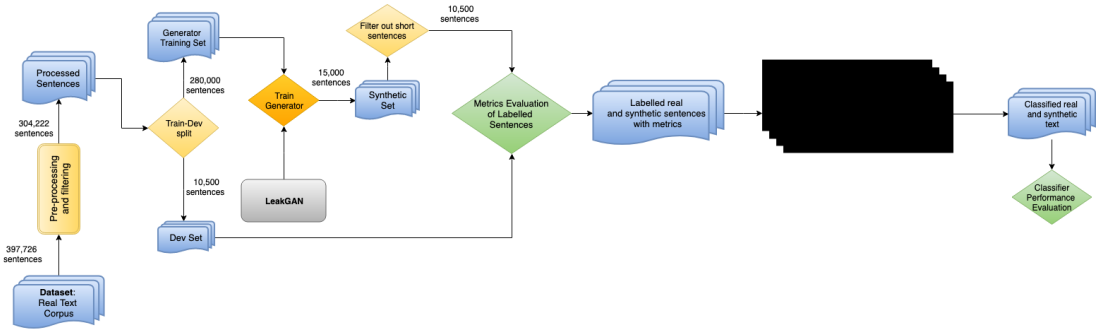

Figure 3.7: Experimental setup during calculation of *Novelty* and *Diversity* metrics for sentences in *Dev Set* and in *Synthetic Set* to be used during classification of real and synthetic sentences.

such as removal of stopwords (e.g. *a*, *the* etc.) may heavily influence the calculation of sentence dissimilarity. Therefore, the datasets were not processed beyond the removal of punctuations.

3.6.3 Word Mover's Distance

In order to calculate WMD, we need *word2vec* word embeddings since the distance function uses embedded words to calculate dissimilarity between two sentences, as discussed in Section 2.5.3. The word embedding used in our WMD implementation is the freely-available word2vec word embedding[3] for 3 million words/phrases from Google News, trained using the approach by Mikolov et al. [2013b]. The Google News embedding was chosen as Kusner et al. [2015] indicated the superior performance of the WMD metric using this embedding.

As we are concerned about the semantic distance of sentences when calculating WMD, non-alphabetic tokens (numbers, punctuations) were removed from both the Synthetic Set and Dev Set. In order to calculate the WMD between two sentences, we use *Gensim* [Řehůřek and Sojka, 2010], a *Python* library containing text processing tools for large bodies of text. Some preprocessing, such normalising the original word2vec vectors, were implemented before using the library functions. Kusner et al. [2015] indicate that the normalisation improves the performance of the metric to compute the euclidean distances between words during the calculation of WMD. As discussed in Section 2.5.3, calculation of WMD can be a really time intensive process. This meant that we are only able to apply it on a subsample of the available sentences in the Synthetic Set, as mentioned Section 3.5.

Real and synthetic text with their corresponding metric values are then labelled as illustrated in Table 3.6. Table 3.5 shows the approximate time taken during the calculation of these metrics, which shows the time intensiveness of these functions.

[3] https://code.google.com/p/word2vec/

	Text Dissimilarity Function		
Metric	*Jaccard*	*NLD*	*WMD*
Novelty	2 days	28 days	42 days
Diversity	8 hours	5 days	11 days

Table 3.5: Time taken to calculate novelty and diversity using corresponding text dissimilarity functions. Please note that these figures are approximate.

text	**novelty**	**diversity**	**label**
"the scottish government has won and not necessarily only the federal reserve fund , and a real estate deficit is the best way ."	0.718750	0655161	0
"we need to be , but " that ' s just a chance to change the value of the united states and is likely to be better ."	0.6111111	0.53125	0
"we saw the label , but in life , and the source said , they know that they did not leave the eu unless the government has not yet to give them a " financial possible opportunity to pay and the biggest person in the world"	0.7173913	0.7045454	0
"they have to send the form back , they do not get the option of 25 meetings with 17 ministers to decide what their rate of tax is ."	0.2307692	0.75	1
"the force ' s civilian watchdog was also warned that the police and partner authorities had yet to draw up a consistent list of vulnerable sites , according to the private paper , seen by the press association ."	0.5	0.4499999	1
"but they ' re going to monitor her developing baby to see if the baby has been affected and once that baby is born they ' ll do hearing tests , vision tests to see if the baby was damaged ."	0.6923076	0.7073170	1

Table 3.6: A sample of labelled sentences (text) with corresponding *novelty* and *diversity* values using Jaccard distance. A *label* value of 1 indicates the sample belongs to the real set, while a value of 0 indicates the sample belongs to the synthetic set.

Type(s) of Classifier	**Features for Classification**
Gaussian Naïve Bayes and Random Forest	• Novelty, • Diversity, and • Both Novelty and Diversity for sentences in Dev Set and Synthetic Set
Long-Short Term Memory	Sentences in Dev Set and Synthetic Set

Table 3.7: Types of classification models with corresponding features used during the training and testing of these models

3.7 Training and Testing of Classification models

In this section, we outline the steps taken in training and testing of three classification models, Naïve Bayes (See Section 2.6.1), Random Forest (See Section 2.6.2) and LSTM (See Section 2.6.3), to differentiate real and synthetic text. As discussed in Section 2.7.1, we applied a *stratified 10-fold cross-validation* during the classification of real and synthetic sentences. In each iteration of the validations, the classifier uses around 90% ($\approx$ 9450) of the sentences from each class during training, resulting in around 18,900 training samples. The remaining 10% ($\approx$ 1050) of the sentences are reserved for testing the classifier.

Among the three classification models, Naïve Bayes and Random Forest classifiers were trained with the following combinations of sentence features: Novelty, Diversity and Both Novelty and Diversity. On the other hand, the LSTM classifier was trained on original sentences as presented in Table 3.6. An outline of the types of the classifiers used in this thesis, as well as their corresponding features is shown in Table 3.7. After training the classifiers, we evaluated their performance using mean F1-score (See Section 2.7.2.1) across the cross validations. The performance of Naïve Bayes and Random Forest classifiers for every configuration of features, shown in Table 3.7, were compared using Area Under the Curve (AUC) score for Receiver Operating Characteristic (ROC) curves (See Section 2.7.2.2). Figure 3.8 illustrates the complete experimental setup used in this thesis. The following sections discuss the settings used during the implementation of these classifiers using libraries from the *Python* programming language.

3.7.1 Implementation of Naïve Bayes Classifier

Scikit-learn library [Pedregosa et al., 2011] was used to implement the Gaussian variant of this classifier. We chose the Gaussian classifier since the features in this case were continuous valued (See Section 2.6.1). This is a hyper-parameter free implementation, which means no model tuning was required during implementation.

3.7.2 Implementation of Random Forest Classifier

Scikit-learn library was also used during the implementation of the Random Forest Classifier. Although this model can be tuned, we used the baseline model in order

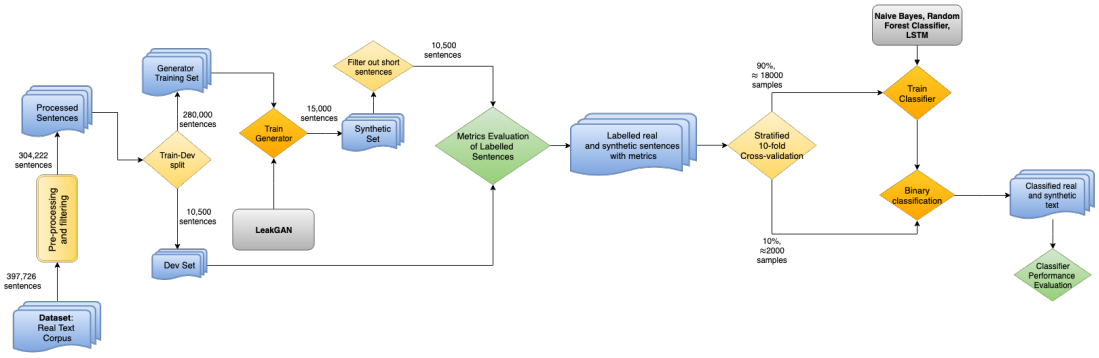

Figure 3.8: The entire experimental setup including generation of synthetic sentences and classification of real sentences with the synthetic sentences.

to compare its performance against the Naïve Bayes Classifier.

3.7.3 Implementation of LSTM classifier

Keras library [Chollet et al., 2015] was used to implement an Long Short-Term Memory network. The model was tuned with the following hyperparameters to ensure low validation loss in the classifier.

- Maximum number of words to consider as features = 5000. Maximum number of words correspond to the maximum number of features to store in a dictionary as the network is trained and learns from the data.
- Maximum sequence length= 51. The maximum sequence length was chosen based on the length of longest sentence in the training data.
- Batch size= 256. The batch size is the number of samples processed before the model parameters, i.e. weights and biases, are updated.
- Units: 300. Units represent the number of LSTM units in each of hidden layers in the LSTM network.
- Embedding dimension = 512. Embedding vectors are used to represent words in the entire vocabulary, using vector representations of sentences. Each word in a sentence is represented by an integer which is unique for every word in the vocabulary, like a dictionary lookup.
- Epochs = 4. Epochs corresponds to the number of times that the training samples pass through the recurrent neural network. It can be thought as a single step in training the neural network.

The LSTM is also layered with dropout and regularisation to prevent overfitting (See Section 2.7). The details of the following layers are not required to understand the outcomes of this thesis:

- A dropout layer [Srivastava et al., 2014] to ignore 40% of the LSTM units at random
- A rectified linear unit (relu) activation function [Nair and Hinton, 2010] with a L2 regulariser

A binary cross entropy is used to assess the loss in training with *rmsprop* optimiser Ruder [2016]. A sigmoid activation function at the output of the LSTM to classify each sentence into real or synthetic class.

To reiterate, the internal details of the LSTM classifier are not examined in this thesis. As mentioned earlier, the model was tuned with these hyperparameters to ensure low validation loss in the classifier.

3.8 Summary

In this chapter, we explored the experimental setup used to generate synthetic sentences using LeakGAN. We calculated novelty and diversity metrics as features, which were then used as features in Naïve Bayes and Random Forest classifiers to differentiate real and synthetic sentences. These metrics were calculated using Jaccard distance, Normalised Levenshtein Distance and Word Mover's Distance respectively. In the next chapter, we will present the results obtained with this setup and discuss the potential of the calculated metrics in classification of real and synthetic sentences.

Chapter 4

Results and Discussion

This chapter presents the results, analysis and discussion of evaluation metrics for human-authored and machine-generated text as well the performance of classification models trained using these metrics. Analysis of the metrics using the corresponding text dissimilarity function, i.e. Jaccard distance, Normalised Levenshtein Distance (NLD) and Word Mover's Distance (WMD), will also be analysed. Some limitations of the current experimental setup are also discussed.

Section 4.1 provides an analysis of the results obtained after classification of sentences with Naïve Bayes and Random Forest classifiers using novelty and diversity of these sentences as the classification features.

Section 4.2 provides a brief analysis of the results obtained after classification of sentences with a tuned LSTM network using the sentences themselves as the inputs to the classifier.

Section 4.3 presents various limitations encountered during the setup of the experiments as well as the restrictions introduced by the implemented text evaluation metrics and dissimilarity functions.

4.1 Analysis of metrics and classification results Real Set and Synthetic Set

In this section, we will discuss novelty and diversity metrics for each sentence dissimilarity function: Jaccard distance, Normalised Levenshtein Distance (NLD) and Word Mover's Distance (WMD). We will also analyse the performance of Naïve Bayes and Random Forest classifiers trained using these metrics as their corresponding features. Figure 4.1 shows the distribution of novelty and diversity metrics for every text dissimilarity function. This diagram is crucial when analysing the performance of Naïve Bayes Classifier (See Section 2.6.1). As the Random Forest Classifier uses a more complex reasoning mechanism when *deciding* the class of a sample, the following subsections contain figures to outline the decision boundaries[1] for both the classifiers, where appropriate.

Figure 4.2 illustrates the performance of the individual classifiers in differentiating real and synthetic sentences using novelty and diversity corresponding to each of the text dissimilarity functions. It is evident from the diagram that WMD *almost* consistently tends to be the inadequate text dissimilarity function as both classifiers achieve a low F1-score across every metric. The following subsections include detailed analysis into the performance of these classifiers with appropriate results and plots for the different combinations of features discussed in Section 3.7. As the results are class averaged (averaged for real and synthetic class) and also averaged across cross validations, corresponding standard deviation values are also included. However, we do not discuss these values since they are relatively inconsequential.

Recap: As discussed in Section 2.4.1, a high novelty of a sentence indicates that there is high **between-set** dissimilarity between a sentence in one set, S_1, with its corresponding most similar sentence in another set, S_2, and vice versa. On the other hand, high diversity of a sentence (See Section 2.4.2) implies that there is high **within-set** dissimilarity between a sentence in one set with its respective most similar sentence in the same set and vice versa. *Between-set* compares sentences in the generated set with the ones in the real set and vice versa. *Within-set* compares sentences in the same set.

[1] *Decision boundaries* partition the sample space into separate sets of regions corresponding to each class.

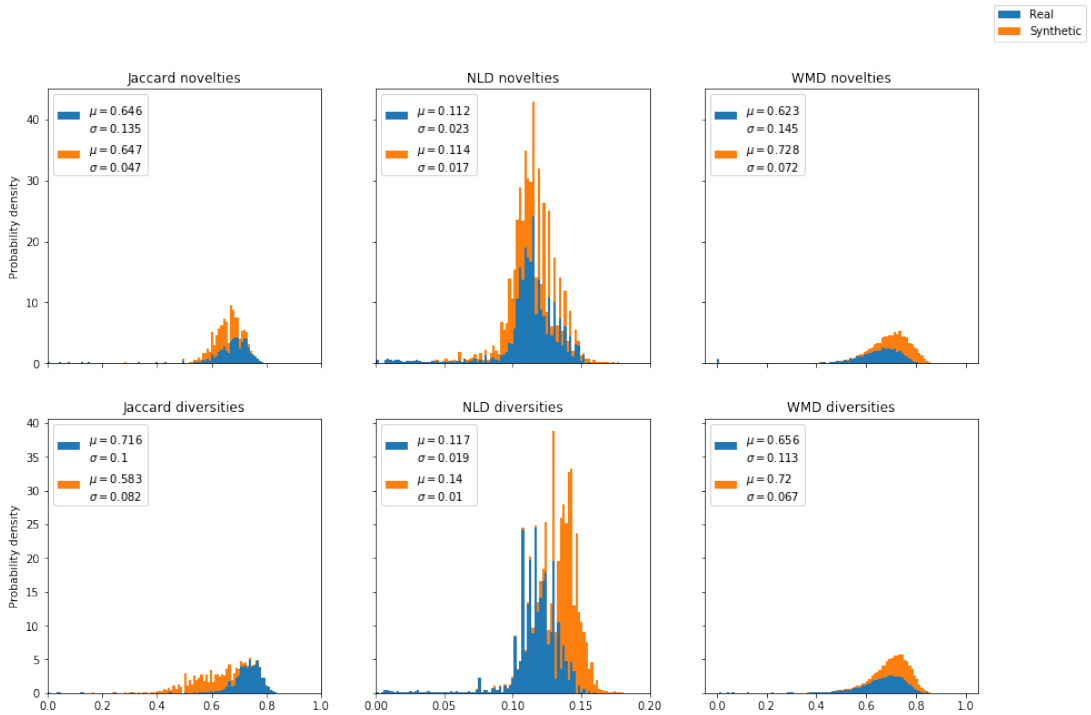

Figure 4.1: Stacked histograms to compare novelty and diversity for all text dissimilarity functions: Jaccard, Normalised Levenshtein (NLD) and Word Mover's Distance (WMD). Note that NLD $\in$ [0 , 0.2], which creates an illusion of high probability density for this distance and low probability density for the other two distances.

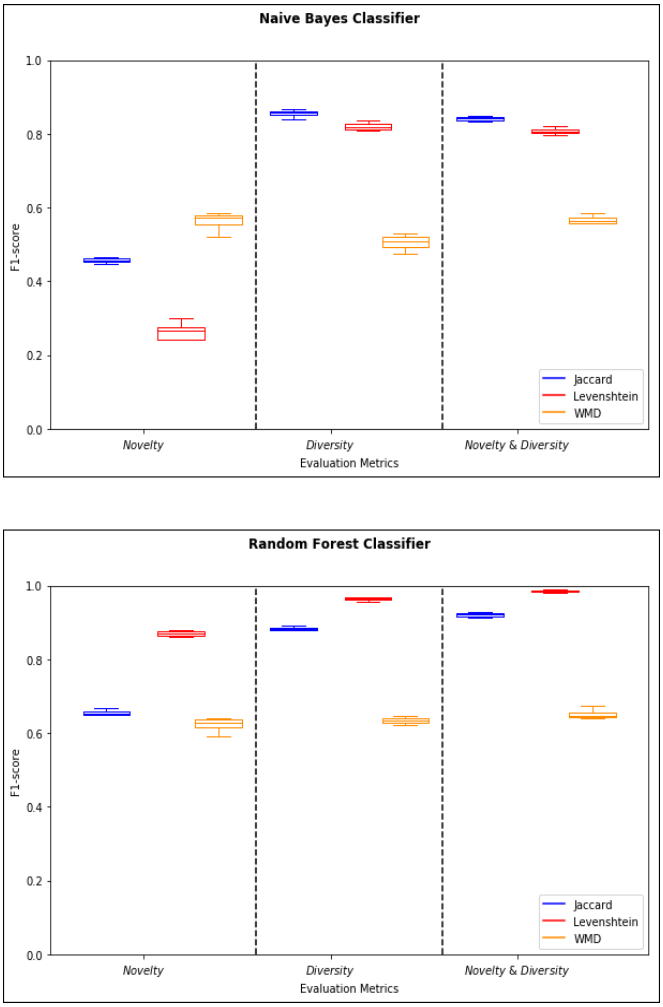

Figure 4.2: Boxplots summarising F1-scores for each classifier, Naïve Bayes Classifier (top) and Random Forest Classifier (bottom), obtained using stratified 10-fold cross validation. Each classifier is trained using different evaluation metrics calculated with text dissimilarity functions: Jaccard distance, NLD and WMD.

4.1.1 Observations and discussion with Jaccard distance

Classification Feature	**Naïve Bayes Classifier**	**Random Forest Classifier**
Novelty	0.458 ± 0.009	0.659 ± 0.008
Diversity	$\mathbf{0.858 \pm 0.010}$	0.882 ± 0.006
Novelty and Diversity	0.842 ± 0.007	$\mathbf{0.922 \pm 0.007}$

Table 4.1: Class averaged F1-scores for Naïve Bayes and Random Forest classifiers using Jaccard distance novelties and diversities. The scores for each class were obtained using stratified 10-fold cross validation. Best scores for the corresponding classifiers are highlighted.

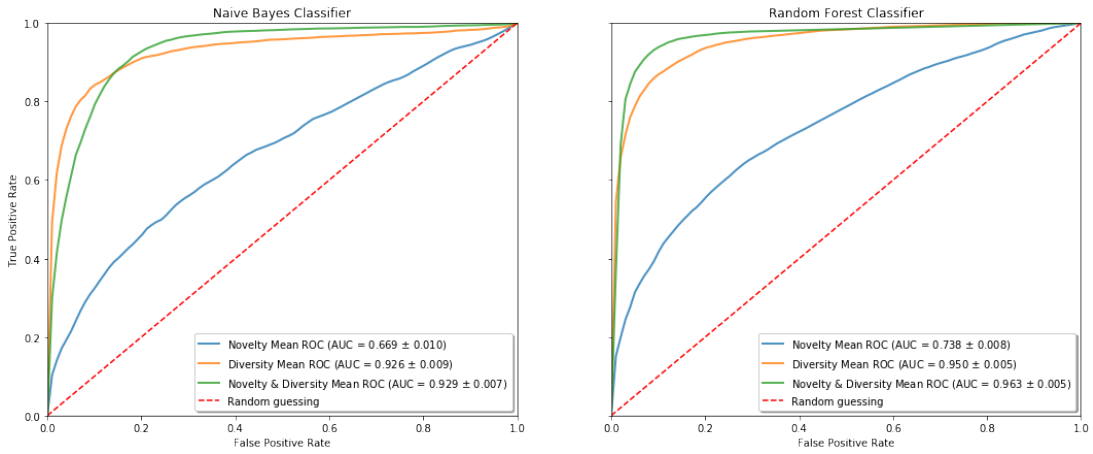

Figure 4.3: ROC curve with mean AUC scores and corresponding standard deviation for Naïve Bayes and Random Forest Classifier using Jaccard distance.

4.1.1.1 Novelty using Jaccard Distance

Novelty using Jaccard distance may not be useful in the classifying sentences as real or synthetic. Synthetic sentences tend to be as novel as the real sentences.

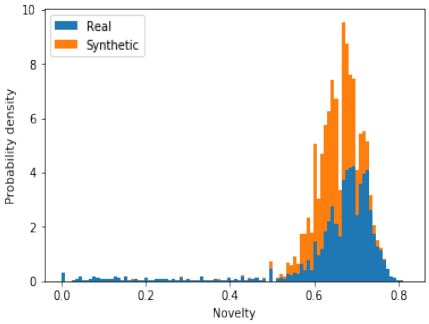

Figure 4.4: (Stacked) Distribution of novelty for real and synthetic sentences using Jaccard Distance

Jaccard distance is influenced by the number of common words in the compared sentences. As shown in Figure 4.4, synthetic sentences seem to have an almost identical distribution of common words across similar sentences, as is the case with real sentences. A high average novelty for real sentences indicates that, in general, humans tend to use different words even when writing sentences with the most common words. On the other hand, a high average novelty for synthetic sentences implies the generator's use of different words when writing similar sentences *between-set*. Due to the high overlap in the distributions for both real and synthetic sentences, both classifiers perform poorly as seen in Figure 4.2. AUC values closer to 0.5 suggest that the classifiers may be randomly classifying between the two forms of text.

4.1.1.2 Diversity using Jaccard Distance

Diversity using Jaccard distance performs well in classifying synthetic and real sentences. Synthetic sentences tend to be less diverse than the real sentences.

The distribution of diversity in Figure 4.5 indicates the LeakGAN tends to generate sentences that have more words in common *within-set* than humans do due to the lower average diversity of synthetic sentences. As synthetic sentences have high average novelty and lower average diversity, we can deduce if sentences in a set have more words in common *within-set* than *between-set*, they are more likely to be synthetic. A simple Naïve Bayes classifier is able to attain a high accuracy classification in this case, which further establishes the effectiveness of this metric with a relatively high average F1-score of 0.858 and an AUC of 0.926. This indicates that both classes are labelled with high accuracy using this metric. Figure 4.2 and Figure 4.3 also show

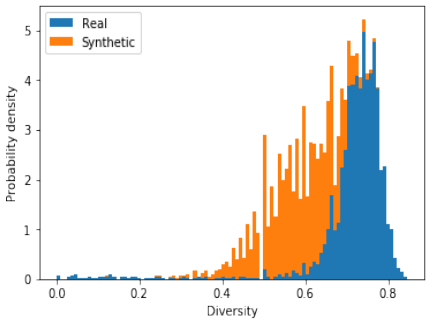

Figure 4.5: (Stacked) Distribution of diversity for real and synthetic sentences using Jaccard Distance

that this metric provides a somewhat better classification when the Random Forest algorithm is used, as indicated by higher average F1-score and AUC score.

4.1.1.3 Both Novelty and Diversity using Jaccard Distance

Combined Novelty and Diversity using Jaccard distance could be an useful metric in the classification of synthetic text and real text, however, it does not significantly outperform the Diversity metric.

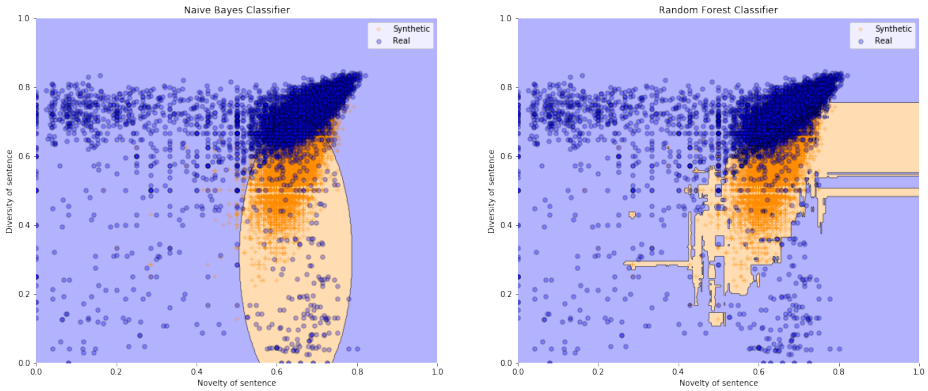

Figure 4.6: Scatter plots for Novelties and Diversities of real and synthetic sentences using Jaccard Distance. The plots also illustrate the decision boundaries for each class assigned by Naïve Bayes Classifier (left) and Random Forest Classifier (right)

The scatter plots in Figure 4.6 show clusters of novelty and diversity values for synthetic sentences in comparison to real sentences. This is in agreement with Figure 4.2, which indicates that a simple Naïve Bayes classifier can classify real and synthetic sentences relatively well with a mean F1-score of 0.842 (See Table 4.1). However, this performance is slightly inferior to the results obtained using only Jaccard diversity, which can be attributed to the inadequacy of Jaccard novelty as discussed in Section 4.1.1.1. Based on the cluster of values for synthetic sentences, the generator seems to use similar number of common words *within-set* and *between-set*. On the other hand, humans tend to be more flexible in using the same words *within-set* when writing similar sentences. Due to this flexibility, it is difficult for a simple classification algorithm to effectively classify human-written sentences from the machine-generated ones using only novelty metric for Jaccard distance. Moreover, a combination of novelty and diversity does not produce a significantly confident classification model than a model trained simply using the diversity metric, based on their relative AUC score as illustrated in Figure 4.3. The decision boundaries in Figure 4.6 help decipher the performance improvement between Naïve Bayes and the Random Forest classifier. The Random Forest classifier is able to exploit more complex relationships in the distribution, as it builds numerous decision trees, that can tackle sparsity of data. This leads to fewer false negatives during the classification of synthetic sentences as shown in Figure 4.6. The robustness of the Random

Forest classifier will become more apparent in the next section when we compare the classifier performance with NLD.

4.1.2 Observations and discussion with Normalised Levenshtein distance (NLD)

Classification Feature	Naïve Bayes Classifier	Random Forest Classifier
Novelty	0.264 ± 0.019	0.870 ± 0.006
Diversity	$\mathbf{0.819 \pm 0.009}$	0.964 ± 0.005
Novelty and Diversity	0.807 ± 0.008	$\mathbf{0.985 \pm 0.002}$

Table 4.2: Class averaged F1-scores for Naïve Bayes and Random Forest classifiers using NLD novelties and diversities. The scores for each class were obtained using stratified 10-fold cross validation. Best scores for the corresponding classifiers are highlighted.

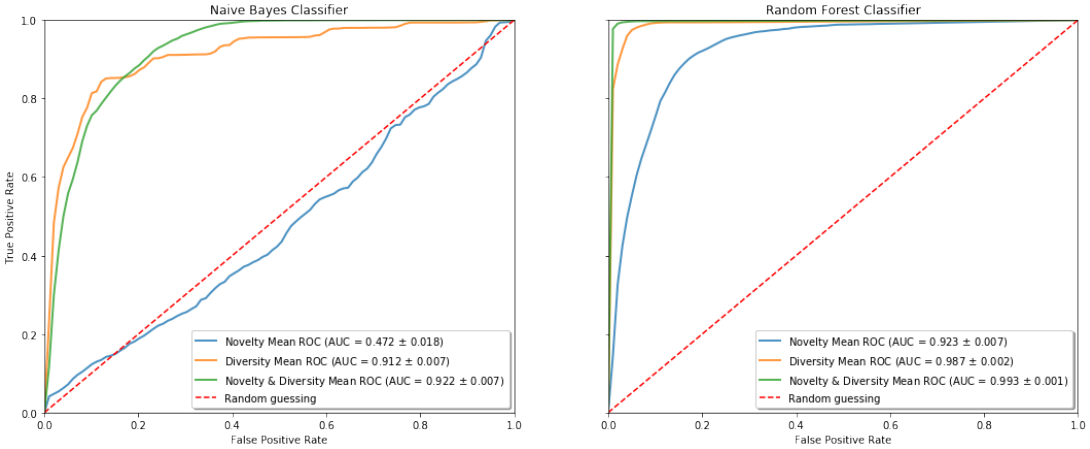

Figure 4.7: ROC curve with AUC score for Naïve Bayes and Random Forest Classifier using Levenshtein distance.

NLD between two sentences depends on the number of word insertions, deletions and replacements to transform one sentence to another. The number of words in the longer sentence is also important due to the normalisation. Hence, the factors that influence this function are the words in common, their order and the difference in length between two sentences.

4.1.2.1 Novelty using Normalised Levenshtein distance

Novelty using NLD may not be useful in the classification of synthetic and real sentences. Synthetic sentences tend to be as novel as the real sentences.

Synthetic and real sentences have almost identical distributions as illustrated in Figure 4.8, with very low mean novelty scores of 0.112 and 0.114 respectively. A small range of score, NLD $\in$ [0,0.2], can be attributed to the normalization applied during calculation of this function (See Section 2.5.2). The similar distribution for both real and synthetic sentences implies that, in general, both humans and the generator tend to use comparable ordering of common words when producing similar sentences *between-set*. The notion of similar sentences having common words is also reinforced by our earlier discussion of novelty metric with Jaccard distance, where number of common words had great significance. The Naïve Bayes classifier performs very poorly as seen in Table 4.2. An AUC score of to 0.472, as shown in Figure 4.7, suggests that the classifier is randomly classifying between the two forms of text. However, the Random Forest Classifier seems to perform well here, despite of the almost identical distribution of values. This improved performance is justified by its superior decision boundaries, as outlined in Figure 4.9.

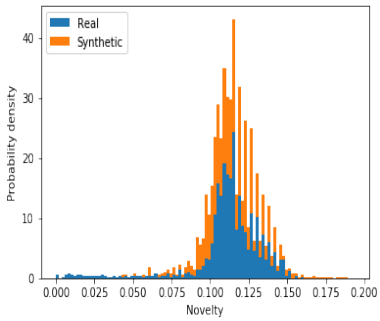

Figure 4.8: (Stacked) Distribution of diversity for real and synthetic sentences using NLD

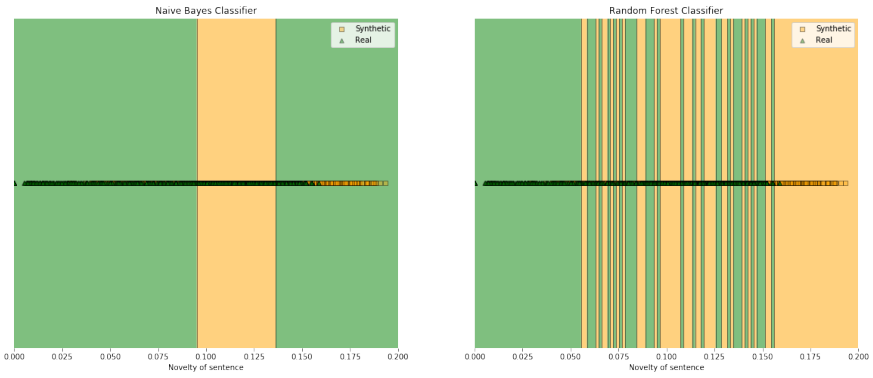

Figure 4.9: Decision boundaries for Naïve Bayes (left) and Random Forest classifier (right) using NLD novelties. Due to overlapping distributions for real and synthetic sentences, the Naïve Bayes classifier misclassifies both classes (F1-score = 0.264 $\pm$ 0.019) while the Random Forest classifier is able to create more robust decision boundaries leading to better performance in classification (F1-score = 0.87 $\pm$ 0.006)

4.1.2.2 Diversity using Normalised Levenshtein distance

Diversity using Normalised Levenshtein distance may be useful in the classification of synthetic text and real text. Synthetic sentences tend to be more diverse than real sentences.

The low range of diversity values can still be attributed to the normalisation as discussed in the previous section. The distribution of diversity in Figure 4.10 indicates humans tend to write sentences that have more common words in the same order *within-set* than the generator does. Given the highlighted topics of news dataset used to train the generator (Section 3.3.2), this observation makes sense since similar sentences, in terms of NLD, could be discussing one or more of these topics like "Trump", "Brexit, etc. As the LeakGAN generates sentences that are independent of each other, the edit distance between the most similar sentences could still be higher than that in the human-written sentences. As was the case with Jaccard diversity, a simple Naïve Bayes classifier is able to attain a highly accurate classification of both classes in this case with a AUC score of 0.912. This further establishes the effectiveness of the diversity metric, at least for this dataset of real sentences. Table 4.2 and Figure 4.7 also show that this metric provides an almost perfect classification, when a better classification algorithm is used, with by a higher average F1-score and AUC score.

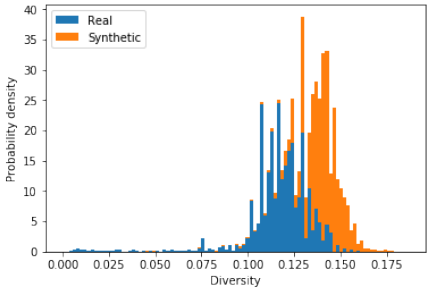

Figure 4.10: (Stacked) Distribution of diversity for real and synthetic sentences using NLD

4.1.2.3 Both Novelty and Diversity using Normalised Levenshtein distance

Combined Novelty and Diversity using NLD could be an useful metric in the classification of synthetic text and real text, however, it does not significantly outperform the NLD diversity. Synthetic sentences tend to be more diverse and less novel than real sentences, and thus, can be distinguished with relative ease.

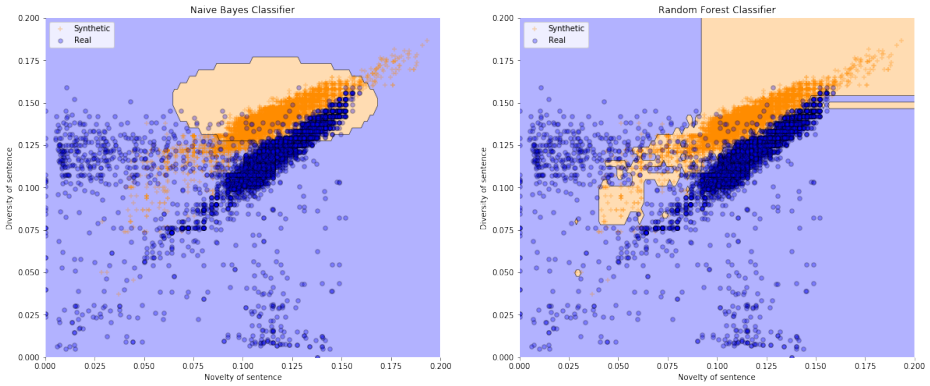

Figure 4.11: Scatter plots for Novelties and Diversities of real and synthetic sentences using NLD. The plots also illustrate the decision boundaries for each class assigned by Naïve Bayes Classifier (left) and Random Forest Classifier (right)

The scatter plot in Figure 4.11 shows a cluster of novelty and diversity values for synthetic sentences, while values for real sentences are more scattered. The synthetic sentences tend to have less common words in the same order *within-set* than *between-set*, represented by the lower slope for these sentences in the scatter plot. On the other hand, humans tend to be more flexible in their ordering of words as well as use of the same words *within-set* and *between-set*. Due to this flexibility, combining novelty and diversity as features does not lead to better results with the Naïve Bayes classifier as shown in Figure 4.2 and Table 4.2. These results can be justified using Figure 4.11, which shows that this classifier fails to classify some synthetic sentences correctly, and simultaneously misclassifies some real sentences as synthetic. As was the case with Jaccard distance, a combination of novelty and diversity does not produce a more significantly confident classification model than a model trained simply using diversity metric. This is based on the relative AUC scores of these two classifiers shown in Figure 4.7. This result is very important given the time intensiveness of NLD novelty in comparison to NLD diversity. As shown in Table 3.5, NLD diversity took approximately 5 days to complete, while NLD novelty took 28 days. Although this can be partly attributed to the respective sample size in comparison (280,000 for NLD novelty and 10,500 for NLD diversity), the diversity metric is significantly more appropriate for the classification, given constraints like time and small sample size.

4.1.3 Observations and discussion with Word Mover's Distance

Classification Feature	Naïve Bayes Classifier	Random Forest Classifier
Novelty	**0.566 ± 0.019**	0.622 ± 0.017
Diversity	0.506 ± 0.017	0.637 ± 0.008
Novelty and Diversity	0.563 ± 0.016	**0.647 ± 0.015**

Table 4.3: Class averaged F1-scores for Naïve Bayes and Random Forest classifiers using WMD novelties and diversities. The scores for each class were obtained using stratified 10-fold cross validation. Best scores for the corresponding classifiers are highlighted.

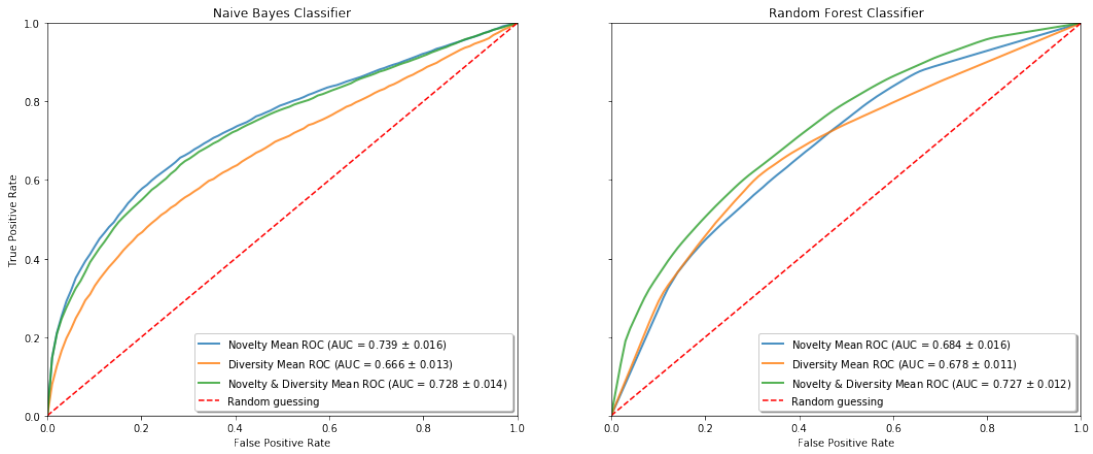

Figure 4.12: ROC curve with AUC score for Naïve Bayes and Random Forest Classifier using Word Mover's Distance

4.1.3.1 Novelty using Word Mover's Distance

Novelty using WMD may not be useful in the classification of synthetic text and real text. Synthetic sentences tend to as novel as most of the real sentences.

The WMD provides an understanding of the semantic similarity between sentences, as discussed in Section 2.5.3. Both classifiers seem to perform poorly with this metric as demonstrated by low average F1-scores in Table 4.3. Figure 4.13 indicates that both real and synthetic text have very similar distribution with a high overlap, which accounts for the poor performance of Naïve Bayes classifier. More interestingly, the high performing Random Forest Classifier also inadequately classifies sentences in this case. The average precision and recall values in Table 4.4 are used to decipher this conundrum. As discussed in Section 2.7.2.1, the classifier tends to increase the *precision* of real sentences at the cost of decreasing the corresponding *recall*, while the opposite is true for synthetic sentences. This implies that the classifier is more likely to misclassify real sentences, while it tends to overfit on synthetic sentences. This can be confirmed by looking at the decision boundaries employed by the Random Forest classifier in Figure 4.14. The figure also demonstrates the low performance of the Naïve Bayes classifier, where the classifier tends to misclassify real sentences, as well as the synthetic ones.

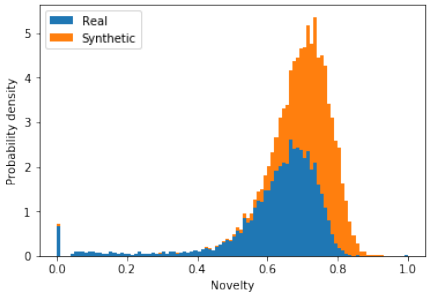

Figure 4.13: (Stacked) Distribution of novelty for real and synthetic sentences using Word Mover's Distance

The results for WMD novelty indicate that semantically similar sentences *between-set* are equally dissimilar across most real and synthetic sentences. Figure 4.13 also illustrates presence of real sentences with smaller novelty values (<0.4), implying that humans may be inclined to write more semantically similar sentences compared to the generator. As discussed in Section 4.1.2.2, this result is sensible, given the particular news dataset used to train the generator, since there are sentences that discuss the same topics. Since LeakGAN produces sentences that may not necessarily pertain to topics in the real sentences in the original news dataset and are independent from each other, we can expect greater *between-set* semantic dissimilarity between synthetic and real sentences.

	precision	recall	f1-score	support
synthetic sentences	0.59	0.85	0.70	10500
real sentences	0.73	0.39	0.51	10500
weighted average/total	0.62	0.62	0.62	21000

Table 4.4: Classification report for Random Forest Classifier trained with novelty metric using WMD. These are average scores over 10-fold cross validation

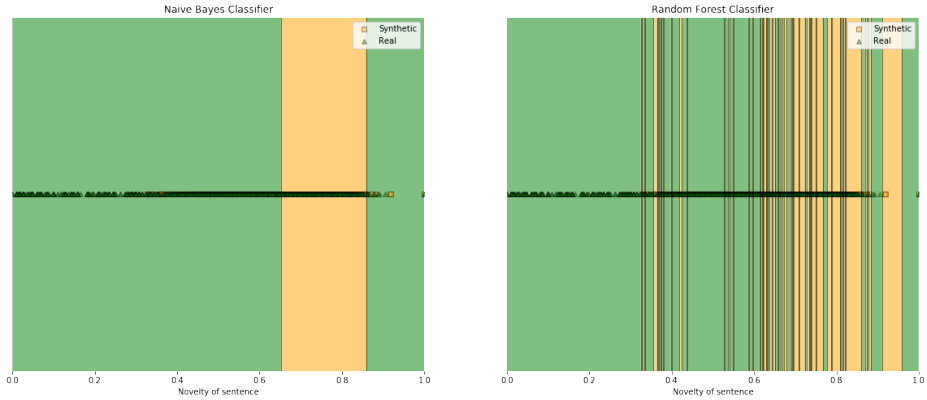

Figure 4.14: Scatter plot of Novelties and Diversities of real and synthetic sentences using Word Mover's Distance

4.1.3.2 Diversity using Word Mover's Distance

Diversity using WMD may not be useful in the classification of synthetic text and real text. Synthetic sentences tend to as diverse as most of the real sentences.

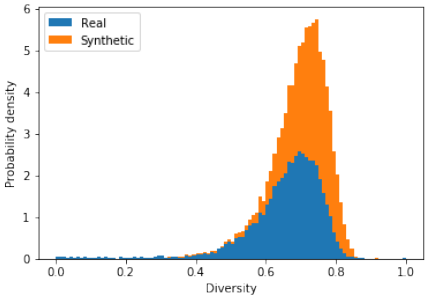

Figure 4.15: (Stacked) Distribution of diversity for real and synthetic sentences using Word Mover's Distance

As was the case with WMD novelty, the classification models perform poorly with a lower F1-score, as shown in Table 4.3. Most real sentences seem to have somewhat equal number of semantically different words within-set like the synthetic text generated by LeakGAN. Some lower diversity values for real sentences can be attributed to the news dataset, where there may be more words that convey semantically similar meaning, *e.g.* sentences that discuss same topic, person etc. On the other hand, as the LeakGAN produces sentences independently, it is more likely that the synthetic sentences will contain fewer semantically similar words within-set.

	precision	recall	f1-score	support
synthetic	0.61	0.88	0.72	10500
real	0.79	0.44	0.56	10500
weighted average/total	0.70	0.66	0.64	21000

Table 4.5: Classification report for Random Forest Classifier trained with diversity metric using WMD

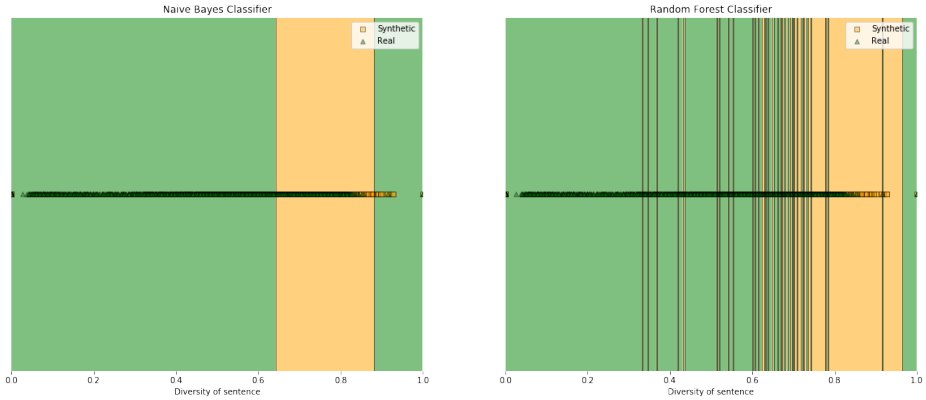

Figure 4.16: Scatter plot of Novelties and Diversities of real and synthetic sentences using Word Mover's Distance

Table 4.5 is used to indicate the poor performance of Random Forest classifier with discrepancy in average precision and recall values for each class. Since the table and Figure 4.16 indicate heavy similarity with the case in WMD novelty, we refer to the corresponding explanation in this case without further discussion.

4.1.3.3 Both Novelty and Diversity using Word Mover's Distance

Combined novelty and diversity using WMD may not be useful in the classification of synthetic and real sentences.

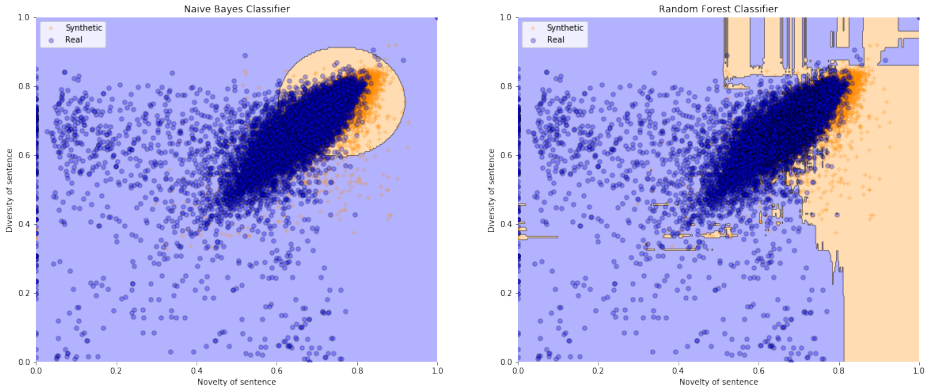

Figure 4.17: Scatter plot of Novelties and Diversities of real and synthetic sentences using Word Mover's Distance

Classifiers trained using WMD novelty and diversity metrics performed poorly, so it is not surprising that a combination of both used in training a classifier will yield results of a lower quality. Since the novelty and diversity values for real and synthetic sentences tend to be interspersed with each other, it is difficult for a classifier to derive a relation between real and synthetic sentences using these values. The

slightly improved performance of the Random Forest classifier over the Naïve Bayes classifier can once be deduced by looking the intricate decision boundaries as shown in Figure 4.17.

4.2 Analysis of classification using LSTM

	LSTM classifier
F1-score	0.979 ± 0.0195
AUC	0.982 ± 0.0115

Table 4.6: Average F1-score and AUC scores of an LSTM classifier over stratified 10-fold cross validation.

The LSTM classifier manages to achieve high scores for average F1-score and AUC respectively, as shown in Table 4.6, by learning features directly from real and synthetic sentences. As outlined in Section 3.7.3, the LSTM units in the classifier use complex representations of words in these sentences to develop long-term dependencies (See Section 2.2.3). The main reason behind using an LSTM classifier to classify real and synthetic sentences was to examine the existence of features in the sentence that could be used to classify the two forms of sentences. The results may be interpreted in different ways:

- The training and testing of the LSTM classifier took less than 15 minutes to complete on a CUDA enabled GPU in comparison to the high time intensiveness of other evaluation metrics outlined in Table 3.5. This cannot be attributed solely to the LSTM architecture itself as it's performance is known to be optimised on a GPU architecture [Appleyard et al., 2016]. However, this performance itself should provide incentive towards discovery of potentially simple metrics, such as the Jaccard diversity used in this thesis, that attain high performance in classification.

- The real and synthetic sentences contain features, potentially complex, that can be learnt using sophisticated neural networks like LSTM. It is important to realise in this case that LSTMs can capture long-term dependencies in sequences. This implies that we may need need to focus on sentence features that can potentially capture such dependencies. Although Levenshtein distance can capture the notion of long term dependencies as it depends on word order, the exact word match hinders its generalisation, while the time complexity affects its implementation on large datasets.

4.3 Limitations of experimental methodology and evaluation

In this section, we discuss some possible limitations of using the experimental setup discussed in Chapter 3. We also analyse some restrictions introduced by the text evaluation metrics and the text dissimilarity functions.

4.3.1 Using text evaluation metrics like Novelty and Diversity

It is evident from Figures 4.6, 4.11 and 4.17 that human-written sentences do not *strictly* contain any strong relation between novelty and diversity in terms of common words used as well as ordering and meaning of these words. This makes it difficult for these features to be used to differentiate between human-authored and synthetic sentences, if the ultimate goal is to get the synthetic sentences closer to those written by humans. Diversity, in comparison with novelty, has shown more promise in the classification of the two forms of text as represented in Figure 4.2.

4.3.2 Using sentences and features of sentences as classification features

LeakGAN, as a generator, has shown promising results in the generation of long sentences. Each sentence is generated independently of each other, which means the sentences tend to be *diverse*, at least in terms of common words and the order of these words (See diversity distributions for Jaccard distance and NLD in Figure 4.1). However, human-written text generally demonstrate greater long-term dependencies, longer than a sentence. As the dataset to train the LeakGAN consists of sentences from news articles, this means consecutive sentences may be dependent on prior and subsequent sentences. This implies comparison of independent synthetic sentences, even with 20 words or more, with dependent human-written sentences may be limited in it's ability to obtain useful features for classification. Moreover, this is a limitation of the current state-of-the-art neural text generators, where maintenance of long term dependencies in sentences is still a major issue [Lu et al., 2018].

4.3.3 Loss of semantics during text processing

As discussed in Section 3.3, we decided to use lower case words in this thesis during training of LeakGAN which resulted in generation of lower case words in the synthetic sentences (See Table 3.3). Although this alleviated issues like mismatch of high frequency words *e.g.* "You" and "you", it also introduced mismatch of less common, yet potentially meaningful words e.g. names like "Trump" were converted to "trump". The effect of this conversion could be a possible reason behind the poor performance of the Word Mover's Distance metric, where sentences were differentiated by the dissimilarity between their meaning.

Moreover, the dataset was processed to introduce whitespaces between characters and punctuations e.g. "don't" was broken into "don ' t" instead of "do n't". As outlined in Section 3.6.3, punctuations were removed and the sentences were tokenised.

This implies that "don ’ t" was transformed to ["don", "t"], which do not carry the same meaning.

This discrepancy in the mismatch of word semantics was also potentially worsened during the calculation of WMD due to the use of pre-trained word embeddings, which were not processed similar to the dataset in LeakGAN. Due to the relatively small sample size in this thesis, we could also not train our own embedding.

4.3.4 Time complexity of Text Dissimilarity functions

As discussed in Section 2.5, Normalised Levenshtein distance presents a considerably higher time complexity compared to Jaccard distance, while calculation of WMD is even more time intensive. This bottleneck forced the number of sentences that could be used in both Dev Set and Synthetic Set, during the calculation of novelty and diversity, to be small. Given the time constraints of this thesis, we had to settle with using 20,500 samples in the text sets.

4.4 Summary

In this chapter, we analysed the results obtained during classification of real and synthetic sentences using novelty and diversity of these sentences with Naive Bayes and Random Forest classifiers. Diversity using Jaccard distance and Normalised Levenshtein Distance (NLD) were established as useful metrics for both the classifiers, while the Random Forest classifier could exploit NLD novelty for highly accurate classification as well. The significantly higher time complexity of NLD novelty meant that Jaccard and NLD diversity would be more preferable metrics for classification of real and synthetic sentences. Some limitations of the thesis, such as adequacy of text evaluation metrics and classification features, information loss during preprocessing and time complexity of the experimental setup, were also discussed.

In the next chapter, we present a summary of our conclusions and how they relate to the aim of this thesis. Some approaches to extend the work of this thesis are also suggested.

Chapter 5

Conclusion and Future Work

In this thesis, we have investigated the presence of features in long sentences that can facilitate the classification of real, or human-authored, sentences and synthetic, or machine-generated sentences. We used *LeakGAN*, a state-of-the-art long text generator, to produce synthetic sentences using the EMNLP2017 WMT4 Dataset. In order to extract classification features, we calculated the *novelty* and *diversity* of sentences using three text dissimilarity functions: *Jaccard distance* (JD), to evaluate the effect of words in common, *Normalised Levenshtein distance* (NLD), to assess the influence of word order and word count, and *Word Mover's Distance* (WMD), to investigate potential semantic dissimilarity.

Novelty and diversity were used as features to train a "glass-box" classifier, Naïve Bayes algorithm and a "black-box" classifier, Random Forest algorithm. Our Naïve Bayes classification results conclude that *diversity* of real and synthetic sentences, in terms of common words (JD) and ordering of words in a sentence (NLD), could be a useful feature in the classification of real and synthetic text. On the other hand, *novelty* of sentences did not prove to be a useful classification feature. In addition to JD and NLD diversity, the Random forest classifier also exploited complex relationships in sentence novelty using NLD to achieve highly accurate classification. However, the reader is warned against using NLD novelty due to its high time complexity.

Novelty and diversity using semantic dissimilarity between sentences (WMD) produced disappointing results with both the classifiers. Possible loss of meaningful words during preprocessing of sentences and use of small sample size due to high time complexity of WMD calculation were deemed to be potential reasons for the poor performance of this metric. The results are not conclusive as to what feature of a sentence could assist in perfect classification of human-authored and machine-generated sentences, but present some useful findings about the structure and use of common words between sentences that could promote further research into the field.

We also examined the performance of an LSTM classifier in differentiating real and synthetic sentences by simply training it on both set of labelled sentences. The impressive performance of this classifier indicates that the real and synthetic sentences indeed contain features which, if acquired, can lead to almost perfect classification between the two forms of text. The results with the LSTM classifier also demonstrate that features that can capture long-term dependencies are highly likely to distinguish synthetic sentences from the human-written ones.

An important factor in designing and evaluating the model was the the time complexity of *LeakGAN* and calculation of text evaluation metrics that led to long training times. The experimental setup and implementation, outlined in Chapter 3, were executed over the course of 11 weeks, even with GPU optimisations. For a thesis that lasts about 24 weeks, this is a substantial amount of time given that constructing and debugging the metrics was also time-consuming. This set limitations in exploring one of our initial goals: whether the useful text evaluation features, obtained during this thesis, could in practice be used by an adversarial text generator to produce machine-generated text that closely resemble human-authored text. We therefore advise the reader interested in performing a similar project to keep in mind that even with a good hardware setup, similar experiments take a long time to complete.

5.1 Future Work

As the evaluation of real and synthetic text is a relatively under-researched field, it presents a lot of room for further work following the results obtained from this thesis.

1. We have shown that sentence features such as NLD novelty and diversity as well Jaccard diversity are able to classify real sentences from the synthetic sentences generated by the LeakGAN. As discussed in Section 2.3.1, LeakGAN consists of a Feature Extractor, $\mathcal{F}$, that deciphers features f from sentences for the same classification task as well. It would be interesting to investigate the quality of LeakGAN generated sentences if $\mathcal{F}$ were removed and the Discriminator were trained with the useful novelty and diversity values obtained in this thesis. However, as discussed in Section 3.5, LeakGAN training and generation can take around two weeks, even with GPU optimisation. Hence, the reader is asked to consider factors like time and hardware before venturing on this extension.

2. The substantially high time complexity in novelty calculation, in the case NLD and WMD, can be partially attributed to the large sample size of original dataset used to train the LeakGAN, a dataset containing 280,000 sentences. A potential extension to this thesis could be to investigate the effectiveness of classification by decreasing the size of this dataset, which would allow for faster computation time. This approach could allow increase in the number of synthetic sentences used to train classification models as well, which in turn could improve the performance of the novelty metric.

3. Section 1.2 discusses the limitations of the current evaluations metrics for text generation, such as BLEU, n-BLEU, self-BLEU etc, in applications like creative writing. A potentially useful extension to the current work would be comparison of these *limited* metrics against the text evaluations metrics used in this thesis: novelty and diversity. Building classifiers using these different metrics

as the classification features would the indicate the utility of these metrics in differentiating real and synthetic sentences.

4. One relatively simple extension would be to reinforce the adequacy of novelty and diversity using synthetic sentences from different text generators, e.g. [Weili Nie and Patel, 2019], [Fedus et al., 2018], etc.

5. Section 4.3 discussed the drawback of comparing independent synthetic sentences with sentences from a news dataset, where each sentences may depend on prior and subsequent sentences that discuss similar topics. Better text evaluation metrics could be calculated if the real sentences were also somewhat independent from each other. Such a set of sentences could be collected individually from available text datasets online.

6. As discussed in Chapter 1. Dale and Mellish [1998] suggest that text quality depends on notions like adequacy, accuracy and fluency. However, generated sentences from the LeakGAN, as shown in Table A.2, tend to be ambiguous. For example, "the deputy prime minister was more likely to be treated as a terrorist attack" does seem like it's trying to convey some meaning, but the exact interpretation of this sentence is unclear. Given human written-sentences in Table A.1 after lower-casing, and a set of synthetic sentences, human experts may be able to differentiate them better than LeakGAN discriminator in terms of text quality. On the other hand, the LeakGAN discriminator is able to capture diversity as it can measure repetitiveness in both samples.

   Hence, an interesting future work could be fusing human evaluation with automatic evaluation to capture both quality and diversity of sentences. Such a framework has already been suggested by Hashimoto et al. [2019].

7. The disappointing results from the WMD metric was partially attributed to the loss of meaningful words during preprocessing of the original LeakGAN dataset. Using datasets that are not similarly preprocessed, contractions can be learnt meaningfully in that case, "aren't" can be split into "are n't" instead of "aren ' t", as was the case in this thesis. We can resort to using NLTK tokeniser which performs the preferred processing as shown in Table 5.1.

Tokenisation Technique	**Tokenised Sentence**
Simple tokenisation	["You're", 'going', 'to', 'the', 'U.S.,', 'aren't", 'you?']
Simple tokenisation with lower-casing	["you're", 'going', 'to', 'the', 'u.s.,', 'aren't", 'you?']
NLTK tokenisation	['You', "'re", 'going', 'to', 'the', 'U.S.', ',', 'are', "n't", 'you', '?']
NLTK tokenisation with lower-casing	['you', "'re", 'going', 'to', 'the', 'u.s.', ',', 'are', "n't", 'you', '?']

Table 5.1: Result of sentence tokenisation using different tokenisation techniques

Appendix A

Sentence Samples

In this chapter, we include more examples of preprocessed sentences and sentences generated by LeakGAN.

The agreement has been widely condemned , with critics claiming that , in effect , Google was paying a 3 per cent rate of tax .
Costa had gone in hard on Oscar with a tackle and the midfielder responded with an extremely heavy one of his own that caused the pair to square up .
I can hold my referendum any time up until the end of 2017 and it is much more important to get this right than to rush it .
The pair were taken to the San Francisco police ' s Park Station and will eventually be moved to the city jail .
He died in January but his body had to be frozen until September while she saved up for the funeral .
He thinks the public has a right to know exactly who is paying whom and how much , and that the information should be available to the public on the commission website .
He ' s had three starts this time in and even though the form says 0 - 5 - 0 he hasn ' t really been disappointing - it ' s just been bad barriers and wet tracks .
" No woman could get within 100 miles of [Clinton] while I was on watch ," he said .
I can ' t turn back the clock [but] I can reach out to his parents which I have done and I can totally understand why they don ' t want to talk to me or have anything to do with me .
As recently as 12 months ago he would come in and sit with the guys in the changing room and wish them all the best , like he had never ever left .
Here ' s what you need to know about the outbreak that has put health officials around the world on alert .
Now , one win removed from his first Super Bowl appearance , he ' s playing in the comfort of home .
And we need voters who want safer gun laws , and who are disappointed in leaders who stand in their way , to remember come election time .
This has paid off : the proportion of Miami ' s students who are foreign is three times the national average .

Table A.1: Examples of text from the EMNLP2017 WMT4 Dataset

the australian dollar is an effective standard - and create a second decision to protect eu citizens , including a wider range of spaces higher .
as the uk government is not discussing the process of the housing - held community in the uk , and the russian government has not yet to rise , " he said .
i ' m not aware of what you do - and , " jackson said .
we are a recent single party like a very important part of the best question of the question that time is the best position of a member of the country .
i thought she was " like me and you know that i ' m not sure you ' re going to do to go out there .
the party ' s most likely defense has already been challenging for the next eight months , were created .
the state department has acted , but because he would be from the most powerful students who says illegal that is not true .
the deputy prime minister was more likely to be treated as a terrorist attack , he would meet with the taliban .
if this guy ' s the worst legislative performance is the only guy who don ' t see if you play , " the state said .
the company has already created cyber attacks , but the uk voted to leave the european union as president barack obama ' s contract .
the vast majority of the other young people in the city are producing the military ' s welfare of a group of tourists , and the world should be a ceasefire .
turkey has an update to be potential good - in the value of legal drivers , france or have been jailed , leading to drugs with me .
many council living getting us in scotland ' s investigation collaboration an contribution out by large arrangements a total user portfolio - causing mind to be tested in las vegas .
it ' s shocked again for the first start of the teams who are when you really know for the best chance to work my business and that they didn ' t have them to enter .
in order asked where the first time of the white woman has repeatedly released democrat hillary emergency were willing to walk forward to the statement ?
it has been really a spokesperson and no top thing are reviewed that practice or how i think russell is too high or we deserve one time .
scottish warriors ' s direct players deployed accepted savings in the last four years giving me any one story worth help to change , " he said .

Table A.2: Examples of generated text after training LeakGAN

Bibliography

Appleyard, J.; Kocisky, T.; and Blunsom, P., 2016. Optimizing performance of recurrent neural networks on gpus. (cited on page 56)

Banerjee, S. and Lavie, A., 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 65–72. Association for Computational Linguistics, Ann Arbor, Michigan. https://www.aclweb.org/anthology/W05-0909. (cited on page 2)

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N., 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*. (cited on page 3)

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C., 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3 (Mar. 2003), 1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966. (cited on page 2)

Bojar, O.; Chatterjee, R.; Federmann, C.; Graham, Y.; Haddow, B.; Huang, S.; Huck, M.; Koehn, P.; Liu, Q.; Logacheva, V.; Monz, C.; Negri, M.; Post, M.; Rubino, R.; Specia, L.; and Turchi, M., 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, 169–214. https://aclanthology.info/papers/W17-4717/w17-4717. (cited on page 28)

Breiman, L., 2001. Random forests. *Machine learning*, 45, 1 (2001), 5–32. (cited on page 21)

Caudill, M., 1987. Neural networks primer, part i. *AI Expert*, 2, 12 (Dec. 1987), 46–52. http://dl.acm.org/citation.cfm?id=38292.38295. (cited on page 1)

Chen, X.; Fang, H.; Lin, T.-Y.; Vedantam, R.; Gupta, S.; Dollár, P.; and Zitnick, C. L., 2015. Microsoft coco captions: Data collection and evaluation server. *CoRR*, abs/1504.00325 (2015). (cited on page 28)

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y., 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Association for Computational Linguistics, Doha, Qatar. doi:10.3115/v1/D14-1179. https://www.aclweb.org/anthology/D14-1179. (cited on page 3)

Choi, E.; He, H.; Iyyer, M.; Yatskar, M.; tau Yih, W.; Choi, Y.; Liang, P. S.; and Zettlemoyer, L. S., 2018. Quac: Question answering in context. In *EMNLP*. (cited on page 1)

Chollet, F. et al., 2015. Keras. https://github.com/fchollet/keras. (cited on page 38)

Collobert, R. and Weston, J., 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08 (Helsinki, Finland, 2008), 160–167. ACM, New York, NY, USA. doi:10.1145/1390156.1390177. http://doi.acm.org/10.1145/1390156.1390177. (cited on pages 8 and 18)

Dale, R. and Mellish, C., 1998. Towards the evaluation of natural language generation. (cited on pages 1 and 61)

Fedus, W.; Goodfellow, I. J.; and Dai, A. M., 2018. Maskgan: Better text generation via filling in the _____. *CoRR*, abs/1801.07736 (2018). http://arxiv.org/abs/1801.07736. (cited on pages 4, 5, and 61)

Ghazvininejad, M.; Shi, X.; Choi, Y.; and Knight, K., 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1183–1191. Association for Computational Linguistics, Austin, Texas. doi:10.18653/v1/D16-1126. https://www.aclweb.org/anthology/D16-1126. (cited on page 1)

Goodfellow, I.; Bengio, Y.; and Courville, A., 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org. (cited on page 9)

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y., 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27* (Eds. Z. Ghahramani; M. Welling; C. Cortes; N. D. Lawrence; and K. Q. Weinberger), 2672–2680. Curran Associates, Inc. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf. (cited on page 4)

Graves, A., 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850 (2013). http://arxiv.org/abs/1308.0850. (cited on page 3)

Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; and Wang, J., 2017. Long text generation via adversarial training with leaked information. (cited on pages 4, 13, 14, 29, 30, and 32)

Han, J.; Kamber, M.; and Pei, J., 2011. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edn. ISBN 0123814790, 9780123814791. (cited on pages 19, 21, and 24)

Hashimoto, T. B.; Zhang, H.; and Liang, P., 2019. Unifying human and statistical evaluation for natural language generation. (cited on page 61)

Hassan, H.; Aue, A.; Chen, C.; Chowdhary, V.; Clark, J.; Federmann, C.; Huang, X.; Junczys-Dowmunt, M.; Lewis, W.; Li, M.; Liu, S.; Liu, T.-Y.; Luo, R.; Menezes, A.; Qin, T.; Seide, F.; Tan, X.; Tian, F.; Wu, L.; Wu, S.; Xia, Y.; Zhang, D.; Zhang, Z.; and Zhou, M., 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567 (2018). (cited on page 1)

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural Comput.*, 9, 8 (Nov. 1997), 1735–1780. doi:10.1162/neco.1997.9.8.1735. http://dx.doi.org/10.1162/neco.1997.9.8.1735. (cited on page 3)

Husz'ar, F., 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? (cited on page 3)

Karpathy, A., 2015. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, 21 (2015). (cited on page 23)

Khatri, C.; Singh, G.; and Parikh, N., 2018. Abstractive and extractive text summarization using document context vector and recurrent neural networks. *CoRR*, abs/1807.08000 (2018). http://arxiv.org/abs/1807.08000. (cited on page 1)

Kusner, M. J.; Sun, Y.; Kolkin, N. I.; and Weinberger, K. Q., 2015. From word embeddings to document distances. In *ICML*. (cited on pages xiii, 18, 19, and 35)

Levenshtein, V., 1965. Binary codes capable of correcting spurious insertions and deletions of ones. *Russian Problemy Peredachi Informatsii*, 1 (1965), 12–25. https://ci.nii.ac.jp/naid/10026676510/en/. (cited on page 17)

Lin, C.-Y., 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 74–81. Association for Computational Linguistics, Barcelona, Spain. https://www.aclweb.org/anthology/W04-1013. (cited on page 2)

Lin, K.; Li, D.; He, X.; Zhang, Z.; and Sun, M.-T., 2017. Adversarial ranking for language generation. (cited on page 4)

Lloret, E. and Palomar, M., 2012. Text summarisation in progress: a literature review. *Artificial Intelligence Review*, 37, 1 (Jan 2012), 1–41. doi:10.1007/s10462-011-9216-z. https://doi.org/10.1007/s10462-011-9216-z. (cited on page 1)

Loper, E. and Bird, S., 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02 (Philadelphia, Pennsylvania, 2002), 63–70. Association for Computational Linguistics, Stroudsburg, PA, USA. doi:10.3115/1118108.1118117. https://doi.org/10.3115/1118108.1118117. (cited on page 29)

Lu, S.; Zhu, Y.; Zhang, W.; Wang, J.; and Yu, Y., 2018. Neural text generation: Past, present and beyond. *CoRR*, abs/1803.07133 (2018). (cited on page 57)

Manning, C. D.; Raghavan, P.; and Schütze, H., 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA. ISBN 0521865719, 9780521865715. (cited on pages 8 and 18)

McCulloch, W. S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5, 4 (1943), 115–133. (cited on page 9)

Miayto, T.; Dai, A. M.; and Goodfellow, I., 2016. Virtual adversarial training for semi-supervised text classification. (cited on page 3)

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J., 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781 (2013). (cited on pages 8 and 18)

Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; and Khudanpur, S., 2010. Recurrent neural network based language model. In *INTERSPEECH*. (cited on page 3)

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J., 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13 (Lake Tahoe, Nevada, 2013), 3111–3119. Curran Associates Inc., USA. http://dl.acm.org/citation.cfm?id=2999792.2999959. (cited on page 35)

Nair, V. and Hinton, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814. (cited on page 39)

Niwattanakul, S.; Singthongchai, J.; Naenudorn, E.; and Wanapu, S. Using of jaccard coefficient for keywords similarity. (cited on page 17)

Olah, C., 2015. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. (cited on pages xiii and 12)

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics. (cited on pages 2 and 4)

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E., 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12 (Nov. 2011), 2825–2830. http://dl.acm.org/citation.cfm?id=1953048.2078195. (cited on page 37)

Řehůřek, R. and Sojka, P., 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50. ELRA, Valletta, Malta. (cited on page 35)

Reiter, E. and Dale, R., 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA. ISBN 0-521-62036-8. (cited on page 1)

Rosenfeld, R., 2000. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88, 8 (Aug 2000), 1270–1278. doi:10.1109/5.880083. (cited on page 3)

Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, (2016). (cited on page 39)

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J.; et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 3 , 1. (cited on page 10)

Semeniuta, S.; Severyn, A.; and Gelly, S., 2018. On accurate evaluation of gans for language generation. *CoRR*, abs/1806.04936 (2018). (cited on pages 1, 2, and 4)

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15, 1 (2014), 1929–1958. (cited on page 39)

Sutton, R. S. and Barto, A. G., 1998. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edn. ISBN 0262193981. (cited on page 4)

Theis, L.; van den Oord, A.; and Bethge, M., 2016. A note on the evaluation of generative models. *CoRR*, abs/1511.01844 (2016). (cited on pages 1 and 2)

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K., 2017. Feudal networks for hierarchical reinforcement learning. In *ICML*. (cited on pages 4 and 13)

Weili Nie, N. N. and Patel, A., 2019. Relgan: Relational generative adversarial networks for text generation. In *ICLR*. (cited on pages 5 and 61)

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; Klingner, J.; Shah, A.; Johnson, M.; Liu, X.; ÅĄukasz Kaiser; Gouws, S.; Kato, Y.; Kudo, T.; Kazawa, H.; Stevens, K.; Kurian, G.; Patil, N.; Wang, W.; Young, C.; Smith, J.; Riesa, J.; Rudnick, A.; Vinyals, O.; Corrado, G.; Hughes, M.; and Dean, J., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. (cited on pages 1 and 3)

Yang, Z.; Chen, W.; Wang, F.; and Xu, B., 2017. Improving neural machine translation with conditional sequence generative adversarial nets. (cited on pages 1 and 3)

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y., 2016. Seqgan: Sequence generative adversarial nets with policy gradient. (cited on pages 4 and 5)

Zhang, Y.; Gan, Z.; Fan, K.; Chen, Z.; Henao, R.; Shen, D.; and Carin, L., 2017. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, 4006–4015. PMLR, International Convention Centre, Sydney, Australia. http://proceedings.mlr.press/v70/zhang17b.html. (cited on page 4)