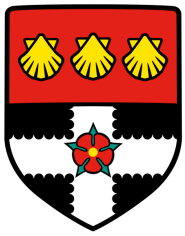

University of Reading

Department of Computer Science

An Analysis of Natural Language Processing Techniques Applied to Generating Tweets

Joseph Rickard

Supervisor: Dr Varun Ojha

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

May 10, 2020

Declaration

I, Joseph Rickard, of the Department of Computer Science, University of Reading, confirm that all the sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Joseph Rickard
May 10, 2020

Abstract

This report investigates the use of recurrent neural network (RNN) models, in particular those composed of gated recurrent unit (GRU) and long short-term memory (LSTM) neurons in text generation tasks. The models are trained on tweets from an individual user which are embedded using the Word2Vec model. Both models are found to be able to produce sufficiently accurate text, supported by standard text evaluation metrics including BLEU, METEOR and ROUGE_L. The LSTM model is found to slightly surpass the GRU model in training speed and final loss of the network. It is found that relatively smaller datasets are not a significant constraint in text generation tasks.

Keywords: Natural Language Processing, GRU, LSTM, Word Embedding, Text Generation

Report's total word count: 11600

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Aims and Objectives . . . 1
- 1.3 Problem Statement . . . 2
- 1.4 Summary of contributions and achievements . . . 2
- 1.5 Organization of the Report . . . 3

2 Literature Review **4**
- 2.1 Natural Language Processing . . . 4
  - 2.1.1 Text Generation . . . 4
  - 2.1.2 Natural Language Understanding . . . 6
  - 2.1.3 Text Classification . . . 7
  - 2.1.4 Information Retrieval . . . 7
- 2.2 Word Embeddings . . . 7
  - 2.2.1 One-hot encoding . . . 8
  - 2.2.2 Continuous-Bag-of-Words Model . . . 8
- 2.3 Tweet Generation . . . 9
- 2.4 Language Evaluation . . . 9
  - 2.4.1 BiLingual Evaluation Understudy (BLEU) . . . 9
  - 2.4.2 Metric for Evaluation of Translation with Explicit ORdering (METEOR) 9
  - 2.4.3 Recall-Oriented Understudy for Gisting Evaluation (ROUGE) . . . 10
- 2.5 Summary . . . 10

3 Methodology **11**
- 3.1 Text Preprocessing Methods . . . 11
  - 3.1.1 Sanitization of the text data . . . 11
  - 3.1.2 Vectorization of the text input into word embeddings . . . 11
  - 3.1.3 Training and Test sets preparation . . . 12
- 3.2 Recurrent Neural Networks . . . 12
  - 3.2.1 Long Short-Term Memory (LSTM) . . . 13
  - 3.2.2 Gated Recurrent Unit (GRU) . . . 13
- 3.3 Implementation details . . . 13
  - 3.3.1 Configuration of the network models . . . 14
  - 3.3.2 Initialization of the training . . . 14
- 3.4 Text Generation . . . 14
- 3.5 Evaluating the generated text . . . 15
- 3.6 Summary . . . 15

4 Results **17**
4.1 Gated Recurrent Unit (GRU) Model Text Generation and Evaluation 17
4.1.1 Text Generated by Network of GRUs 18
4.1.2 Language Evaluation Metrics Applied to GRU Generated Text 20
4.2 Long Short-Term Memory (LSTM) Model Text Generation and Evaluation . . 21
4.2.1 Text Generated by Network of LSTMs 21
4.2.2 Language Evaluation Metrics Applied to LSTM Generated Text . . . 23
4.3 Summary 23

5 Discussion and Analysis **25**
5.1 GRU Tweet Generation 25
5.2 LSTM Tweet Generation 25
5.3 Comparison of approaches 25
5.4 Summary 26

6 Conclusions and Future Work **27**
6.1 Conclusions 27
6.2 Future Work 28

7 Reflection **29**

Chapter 1

Introduction

1.1 Background and Motivation

The field of natural language processing and text generation in particular is an area of constant developments. The proposition of the continuous-bag-of-words (CBOW) model by Mikolov, Chen, Corrado and Dean (2013) provided an opportunity for significant advances in text generation using this technique to retain the context of words whilst creating an efficient representation. Using CBOW, many studies have been done on large datasets for text classification as well as text generation, however there is little research into the effect of relatively small datasets with short strings, particularly in the form of Tweets. This paper aims to investigate how a text generation model can perform in this context.

Gated Recurrent Units (GRU), [a specialised type of Long Short-term Memory network (LSTM) Cho et al. (2014)] have been popularised for language processing tasks due to their increased efficiency but have not been investigated for generating tweets. Comparing the effectively of a network of GRU neurons and a network of standard LSTM neurons will also provide insight into the applications of these techniques.

1.2 Aims and Objectives

This report aims to investigate to what extent a GRU or a more traditional LSTM network can be trained using a CBOW model to generate tweets, given the tweets of an individual as input. Through this, the investigation will include the preprocessing and data preparation as necessary into a format which will be fed into a recurrent network. Training the network for varying number of iterations affects the accuracy of the output, with underfitting and overfitting being a necessary consideration. The end objective is to ultimately build a model which can consistently generate tweets with a certain degree of accuracy. The accuracy of the generated tweets and by extension the model will be assessed to a language evaluation standard. This research work strives to achieve the following aim:

> Investigate and evaluate the capabilities of GRU and LSTM networks to generate text in the form of tweets for an individual user given the users previous tweets as inputs.

To achieve the mentioned aim, the following objectives are set:

1. Construct a neural network implementing GRUs and a network implementing LSTMs to generate tweets using a given users tweets as training data

2. Assess the accuracy of the two models according to a language evaluation standards (BLEU 1-4 - Papineni et al. (2002), METEOR, ROUGE etc.)

3. Evaluate the differences between the models and discuss the strengths and weaknesses of each approach

1.3 Problem Statement

This project aims to embed the textual input data into a form $x \in \mathbb{R}^d$ where d is the input size. The prediction task can be summarised as building following model (Xie, 2017):

$$p(U) = \prod_{t=1}^{S} p(u_t \mid u_{t-1}) \tag{1.1}$$

where $U = (u_1, u_2, u_3, ..., u_n)$ and represents the sequence of embedded values of length n used as training data. This gives us the probability of each word in the input data u_t given the words which came before it $p(u_t \mid u_{t-1})$. This can be applied to the unknown test data to generate text using the words with the highest probability. The accuracy of the model to the training data at each stage is assessed using a cross entropy loss function $\mathcal{L}$:

$$\mathcal{L}(U, p(U)) = -log(p(u_t \mid u_{t-1}) \tag{1.2}$$

The loss function is used to calculate how far the predicted output is from the actual output. This enables the network to recognise where errors lie. Other loss functions include mean squared error (MSE) which averages the squared errors for each point. This has the effect of heavily penalizing when the deviation is larger. Cross-entropy has an even stricter penalisation of miss-classification.

To adjust the weights (and hopefully minimise the objective function of the model) based on the output and error, a backwards propagating gradient descent algorithm is used. A batch gradient descent algorithm calculates the gradient at each point based upon a calculation using the whole dataset. A stochastic gradient descent algorithm calculates gradients using smaller samples of the dataset instead. This generally results in fewer epochs required to reach the same point as a batch descent algorithm, allowing us to perform more iterations and achieve better results. For this reason, a stochastic gradient descent algorithm is used for this task.

A few considerations of a gradient descent algorithm include avoiding local minima and learning rate. This project utilises the Adam optimizer implemented by Keras[1] which has an adaptive learning rate which can help to avoid local minima and is particularly computationally efficient (Kingma and Ba, 2014).

1.4 Summary of contributions and achievements

This study successfully generates text in the form of tweets for an individual user using both a model constructed of gated recurrent units (GRU), and a model constructed of long short-term memory (LSTM) units. The LSTM model is able to achieve a slightly better performance in terms of training rate as well as producing slightly more accurate text when compared to the input corpus.

The conclusion is also met that text generation using this method is able to be performed using relatively small input datasets, with considerations made for future work exploring this further.

[1] https://keras.io/optimizers/ (accessed on 31 Mar. 2020)

1.5 Organization of the Report

This report breaks down the study into a few sections, starting with a review of the existing literature in the field of natural language processing and more specifically text generation using neural networks. An analysis of the existing literature provides insight into the gaps in the publications and is used to steer the topics investigated in this study. The methodology comes next and breaks down the process by which the investigation was implemented and the decisions made at each stage. A careful critique of ones methodology can highlight improvements and areas which need further study.

The results are presented as the tweets which are output from the model. This shows a clear correlation between the performance of the networks and they output the produce. Alongside the generated tweets, the state of the model at each stage is displayed through considering the loss at each epoch. The resulting scores for an array of evaluation metrics is provided, the implications of which are explored in the discussion and analysis chapter. The discussion chapter also explores the differences between the performance of the two models, as well as the text they generated.

The conclusion provides an opportunity to justify the results seen as well as consider to what extent the report manages to adhere to the objectives set out earlier in this chapter. A reflective summary is put forward to review the process and consider the hurdles faced and overcome.

Chapter 2

Literature Review

2.1 Natural Language Processing

This report explores the use of Natural Language Processing (NLP) techniques to attempt to predict tweets on a given topic or for a particular user. NLP is a field of artificial intelligence and machine learning concerned with *'the application of computational techniques to the analysis and synthesis of natural language and speech' Natural Language Processing: Definition of Natural Language Processing by Lexico* (2020).

Natural language processing fundamentally covers four application areas; text generation, text classification, natural language understanding and information retrieval.

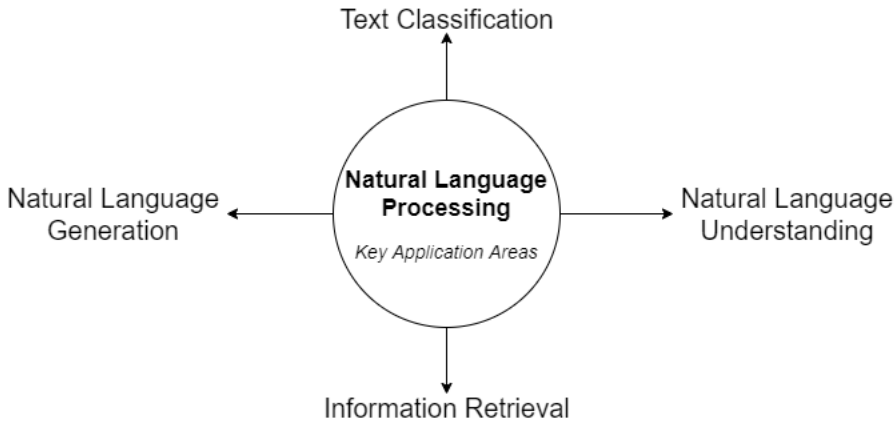

Figure 2.1: Visualisation of fundamental areas in natural language processing. Adapted from lecture notes provided by Ojha (2020)

2.1.1 Text Generation

Text generation (also known as natural language generation or NLG) is one of the key areas within NLP and is the main focus of this report. It consists of the process for creating text so as that it is legible by a human and could be considered to be written by a human. NLG is often performed by defining grammars and context to ensure the sentences generated are grammatically sound (Reiter and Dale, 1997). These grammars are then applied to a known vocabulary to generate text for a specific system. Example of common systems which implement these type of algorithms often include producing weather forecasts (Goldberg et al., 1994) and automated letter generation (Springer et al., 1991).

An alternative method for generating text involves training some kind of predictive model on a preexisting dataset which is then used to generate the text based on what it has learned. This is the type of text generation explored in this work. Neural network models can be applied to a multitude natural language processing tasks, and have been successful in many (Goldberg, 2016) due to their ability to learn and perform on relatively sparse data and data with high dimensionality.

Recurrent Neural Networks

Recurrent neural networks (RNN) are a type of neural network which are implemented by cycling through the nodes with the output of one iteration used as an input towards later iterations. Due to this use of the previous results, RNN models are considered to have a form of 'short-term memory' (Hochreiter and Schmidhuber, 1997). This makes recurrent networks particularly useful and effective in sequential tasks[1], including forecasting and sequence classification as well as many natural language processing applications.

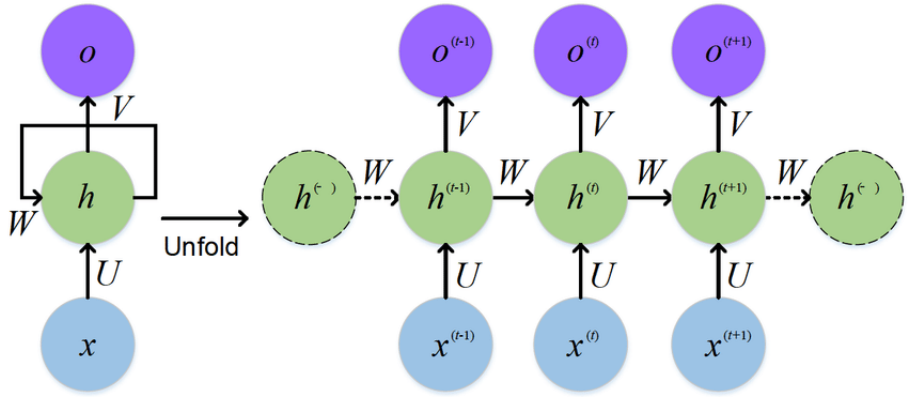

Figure 2.2: Diagram showing the structure of an RNN and how the recursion can be visualised by unfolding (Feng et al., 2017)

Recurrent neural networks, and their variations, are often used for natural language processing tasks (Mikolov et al., 2010) in part due to their innate ability to model and learn complex systems, but also due to the awareness of previous iterations which is fundamentally built into networks of this design.

Many variations on the traditional recurrent neural networks have been proposed which can offer improvements and advantages in particular application areas. One of the most prevalent alternative recurrent network developments is the Long Short-Term Memory RNN proposed by Hochreiter and Schmidhuber (1997).

Long Short-Term Memory The Long Short-Term Memory recurrent neural network (LSTM) was proposed by Hochreiter and Schmidhuber (1997) in their seminal paper "LSTM" as a new type of RNN aimed at resolving some of the traditional RNNs shortcomings. A traditional RNN has a relatively small memory of previous iterations making it unable to "learn to look far back into the past"[1]. Hochreiter and Schmidhuber (1997) criticized the issues with the traditional RNN and highlighted how the back propagation of errors can cause problems such as "oscillating weights" and the tendency for the errors to "blow up or vanish" causing the learning to take significantly longer in some scenarios.

[1]Recurrent Neural Networks (Schmidhuber, 2017)

LSTMs implement a gradient based algorithm for error calculation which, over many iterations, can resolve many of these issues. These changes made the LSTM less susceptible to noisy inputs and reduced the chance for oscillating weights allowing the LSTM able to solve many "previously unlearnable tasks" including the following five (Schmidhuber, 2017) [1]:

1. Recognition of temporally extended patterns in noisy input sequences
2. Recognition of the temporal order of widely separated events in noisy input streams
3. Extraction of information conveyed by the temporal distance between events
4. Stable generation of precisely timed rhythms, smooth and non-smooth periodic trajectories
5. Robust storage of high-precision real numbers across extended time intervals

These developments to the RNN also makes the LSTM a prime candidate for many natural language processing tasks which rely on extended sequences and often have incredibly noisy input data.

Gated Recurrent Units Gated Recurrent Units (GRU) are another advancement from the recurrent neural network working from the basis of an LSTM model. Jozefowicz et al. (2015) investigated whether the LSTM network was an optimal method for modelling sequences, and found that in many situations, the additional parameters and gates built into the LSTM did not provide significant improvements in accuracy. When some of these gates in the LSTM are combined, resulting in the GRU, accuracy was still comparable, but performance was better due to the fewer parameters and weights needing to be updated per iteration. Greff et al. (2016) found that many of the variants of the LSTM, including the GRU, were unable to significantly improve on the architecture when tested on an array of tasks. Chung et al. (2014) found similarly mixed results between the LSTM and GRU but did note that the GRU model performed surprisingly well on language processing tasks, and outperformed on many others.

Feedforward Neural Network

Created in advance of the concept of a recurrent neural network, a feedforward neural network (or FNN) differs in that the graph structure of the network does not include a cycle. This is shown in figure 2.3 which outlines the key differences between the structure of the two variations. The FNN, considered by Schmidhuber (2014) to be one of the first neural networks, created and provided a foundation for further advancements and investigations in machine learning. Whilst FNN models have been used for some natural language processing experiments (Goldberg, 2016), they are generally unable to achieve the same levels of accuracy as the depth of a large RNN can provide (Botha et al., 2017). However, Botha et al. (2017) discovered that despite the trade off in accuracy, a small feedforward network can prove useful in "resource-constrained environments" such as mobile devices and other systems with small amounts of memory available.

2.1.2 Natural Language Understanding

Natural language understanding is the field within natural language processing that covers the topics of sentiment analysis and emotion detection. It is often used to help interpret the meaning from a given text and convert largely unstructured data into a format which can be useful to gauge customer feelings towards a product or a communities opinion on a subject.

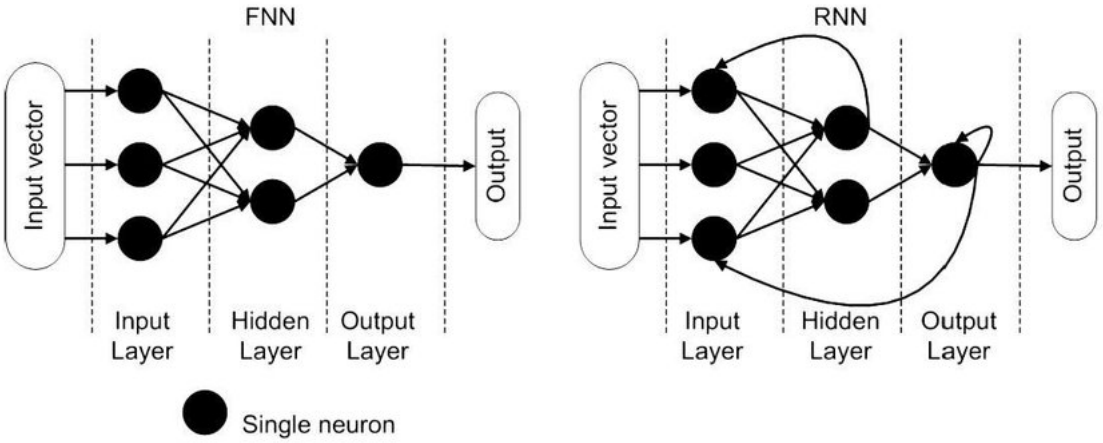

Figure 2.3: Diagram to highlight the differences in the general structure of a feedforward neural network compared to a recurrent neural network (Rezaie, 2015)

2.1.3 Text Classification

Text classification consists of the process of assigning text to a set of nominal categories, usually as a step towards statistical analysis of the data (Zhang et al., 2015). The steps towards this usually consist of applying the natural language processing techniques to extract features from the text. These features can be used as attributes and clustering can be performed on them. Many clustering algorithms exist and are valid choices. A common algorithm applied is the "K-Nearest Neighbour" algorithm which assigns a set number of points with the shortest distance to a central point as the same class. This distance is usually a Euclidean distance for continuous data, but in the case of textual data, where numerical data is not always available, the Hamming distance between strings is also sometimes used.

2.1.4 Information Retrieval

Information retrieval is the final major topic within natural language processing. It is generally defined as the process of searching for and gathering data from a set of documents in order to fulfil a request (Voorhees, 1999). This can be done through literal string searches, or in more advanced scenarios using the semantic meaning of the text in question, to return documents relevant to the user. The advantage of a natural language processing system for information retrieval over the more traditional database search approach is the move away from the requirements to have the data in a formal structure (Voorhees, 1999). NLP information retrieval systems can create semantic summaries of the documents in the search space, and use these to find the document which best matches the users request. In many cases this can return superior matches for the user since the meaning of the request rather than the exact words are matched instead. Beyond the initial generation of the synopses, this form of information retrieval is also able to be significantly more performant by reducing the size of the search space from the vocab size of the documents, to a shorter, more condensed semantic summary.

2.2 Word Embeddings

Word embedding is the converting of text data into a representation which can be better used in a natural language processing task. It aims to create a representation where 'words with

similar meaning have a similar representation' (Brownlee, 2017). There are multiple different ways to achieve this. One of the earlier techniques was the high dimensional sparse method of one-hot encoding. This has since been improved upon, including one of the favoured techniques amongst natural language processing researches of the continuous-bag-of words model, proposed by (Mikolov, Chen, Corrado and Dean, 2013)

2.2.1 One-hot encoding

As mentioned above, one-hot encoding is not the most efficient technique for word embedding but is worth mentioning to understand how it was improved upon in the development of later methods. It entails the use of sparse vectors created for each unique word in the documents vocab (see figure 2.4 below). This does result in a lower dimension representation than the original data, but the dimension is dependent on the size of the vocab. Each word is represented independently, causing the context of the words to be completely lost. This is a particular issue for considering its use in text generation tasks where the context is often vital to generate plausible language.

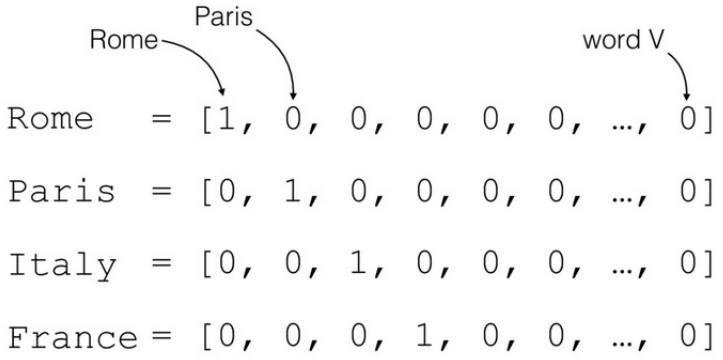

Figure 2.4: Diagram showing an example of how words can be simply encoded using on-hot-encoding (Alvarado et al., 2018)

2.2.2 Continuous-Bag-of-Words Model

One of the challenges of for word embedding with respect to natural language processing is finding a suitable means of representing the data whilst retaining the context of words. The continuous-bag-of-words (CBOW) model was initially proposed in the seminal paper Efficient Estimation of Word Representations in Vector Space (Mikolov, Chen, Corrado and Dean, 2013). It highlights a proposed method for representing text data in an unordered state where context of words is retained. It is a form of word embedding and can thus be frequency based (where each word in the input data is stored alongside the number of occurrences), or prediction based (where each word is stored with the a count occurrences of other words in the same document) Katte (2018). Word2Vec (word to vector) is a method to construct these embeddings Karani (2018), using a recurrent neural network (RNN) to train the context of the words, so that more contextually similar words have closer vector representations. Word2Vec was also originally proposed by Mikolov et al. (Mikolov, Sutskever, Chen, Corrado and Dean, 2013) as an effective use of the CBOW model.

2.3 Tweet Generation

Using these techniques to predict/generate tweets is a largely unexplored area. A lot of work has been done on using data from twitter for sentiment analysis of tweets, for both commercial and academic use, most notably Wakade et al. (2012) who performed text mining and sentiment analysis on Twitter data using the ideas introduced by Pang et al. (2002) in the 2002 seminal paper regarding sentiment analysis using machine learning techniques.

Although these techniques have not been used on data from twitter (for text generation), they have been used however in the more general sense of text learning and generation, mostly on larger blocks of text than the limited 280 characters of Twitter. Subramanian et al. (2018) investigated text generation using a similar approach as this report will take. They used vector representation of their text data and trained a GRU to perform the text generation. The key difference is the vast quantities of data they had at their disposal for the training of their system. The Stanford Natural Language Inference Corpus (SNLI) was just one of the datasets used which contains 570k human-written English sentence pairs (Bowman et al., 2015). The results Subramanian et al. encountered show impressive examples of generated English text. This report will investigate the possibility to generate text by training on a significantly smaller dataset.

2.4 Language Evaluation

An important field tied to the concept of text generation is language evaluation. Without a good metric to assess the quality of the output of natural language models, it becomes very difficult to validate the strengths and weaknesses. There are many different methods introduced for evaluating machine generated text which penalise and reward output strings differently depending on the intended application areas. A few different approaches to evaluating natural language are discussed below.

2.4.1 BiLingual Evaluation Understudy (BLEU)

BLEU is one of the more prominent text evaluation algorithms, originally developed for evaluating translation systems. It was developed by Papineni et al. (2002) and scores the text based upon its similarity to a reference text. The overall belief behind BLEU scoring is 'the closer a machine translation is to a professional human translation, the better it is' (Papineni et al., 2002). The scoring is calculated from the number of occurrences of words from the reference string that appear in the output string. To avoid sentences that use the same words but are not linguistically accurate (see example below as given by Papineni et al. (2002)) scoring highly, the use of n-grams are introduced.

> Candidate: the the the the the the the.
> Reference 1: The cat is on the mat.

BLEU 1-4 is so called because it instead checks for matching n-grams of words of length 1-4 respectively. This results in harsher penalties for sentences like the one mentioned previously, where consecutive words are required to be matching.

2.4.2 Metric for Evaluation of Translation with Explicit ORdering (METEOR)

METEOR is an expansion on the concepts initially raised by BLEU. It aims to overcome some of the downfalls of the BLEU metric by introducing a few additional stages. In addition to the

exact word matching, it also support stemming and synonymy matching. This allows more leniency with alternative words used with the same meaning. This is particularly important for language translation tasks where many words are available for the same meaning. METEOR has been found to outperform BLEU in some language translation evaluation tasks (Denkowski and Lavie, 2014). This is less of an issue for text generation tasks where a set vocab is provided, and the n-gram functionality provided by BLEU can prove more useful.

2.4.3 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

Developed as another metric to asses the accuracy of machine translations and machine generated summaries by Lin (2004), ROUGE investigates the effect of overlapping n-grams and sub-sequences of words. There are several versions of ROUGE, four of which Lin outlines in the paper where he introduces the metric; ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S. ROUGE-N refers to a measure of the overlapping of n-grams from the reference found in the text. ROUGE-L scores highly based upon the longest common subsequence (LCS), making it particularly versatile. ROUGE-W expands on this, favouring consecutive LCS by using weights. ROUGE-S searches for occurrences of skip-bigrams which are present in the reference and hypothesis text.

ROUGE and its derivatives have become invaluable in building and evaluating summarizing tools (Lin and Och, 2004) as well as becoming standards for many other language generation applications.

2.5 Summary

The existing literature, shows an expanse of investigations into the optimal techniques and algorithms for various natural language processing tasks. This report explores an alternative side to this in that it attempts to apply these techniques to a very specific dataset in the form of an individual users tweets. It also investigates applying two different implementations of RNNs to the problem of text generation, allowing a direct comparison of their performance on the same task.

The evaluation metrics are also explored and how they can be of use for text generation tasks. Many evaluation techniques compared alongside each other provide insight into their strengths and weaknesses for this application.

By performing text generation on an individuals tweets, this will also provide some further insight into the capabilities of existing NLP techniques. This includes how well they can replicate the personalised nature that is innate to the tweets of an individual. Additionally, the use of tweets from a single user automatically drastically reduces the input data size, meaning there is fewer data points available to construct the model. If the model is still able to successfully and accurately generate text, this could have significant implications for the requirements needed for further studies and applications of text generation.

Chapter 3

Methodology

The process of building and training a model to generate text has several stages. Preprocessing is an essential step to ensure that the data is in the most efficient format to allow for accurate training and generation. Following that, the model is constructed and the hyper-parameters set. The training consists of the input data being fed into the model over many iterations. The model should then be able to generate text based on an input starting string.

3.1 Text Preprocessing Methods

The first step of the experiment is to gather the data. This task uses data from twitter which is collated using the publicly available developer API (*Twitter Developer API*, 2020). The data from an individual user is to be used as the input to the model, so the entire tweet history for a single account is retrieved. Before this Twitter data is in a state which we can use, a few steps must first be performed.

3.1.1 Sanitization of the text data

To get the text data in a format which can be properly used, the input must be initially sanitized. This consists of cleaning the text to remove unwanted characters and sets of characters which could cause issues when building and training the network model. The first sanitization step that is performed is the removal of hyperlinks from the tweets. These are unlikely to continue to function in the output tweets so they are removed at this stage using a regular expression search. For the most part, the hyperlinks in tweets are references to other tweets so do not pose any use for our requirements anyway.

As well as removing the hyperlinks, the text is also converted entirely into lower case. This is to reduce the vocab size by ensuring identical words with different capitalisation (i.e. "the" and "The") are not represented as different words.

3.1.2 Vectorization of the text input into word embeddings

Vectorization is the process of transforming the text data into a numerical format which can be used in training of the model. The Word2Vec embedding model (NSS, 2017) is used for this step. The set of unique words in the input data are each assigned a unique value. This vector which transforms words to values is retained. The Word2Vec method implements the continuous-bag-of-words model originally proposed by Mikolov, Chen, Corrado and Dean (2013). It allows each sentence in the training data to be represented as a vector of the length of the vocab of the document. It also ensures the context of the words is retained during the

embedding process. This is then used to create a vector representation of the entire input data.

3.1.3 Training and Test sets preparation

The large data is split into smaller data sets which can be fed into the RNN. The data is split into sequences of a set length, with each following sequence being shifted across by one element. For example: sequence 1: {'this', 'is', 'an', 'example'} sequence 2: {'is', 'an', 'example', 'sequence'} These sequences are then passed into the RNN in batches. The first sequence, is used to try and predict the second sequence, and the model weights within the model are adjusted to fit this as correctly as possible.

3.2 Recurrent Neural Networks

This study investigates the use of recurrent networks (RNN) in the field of text generation, in particular, the GRU and the LSTM models. RNNs have been often used in text generation tasks (as outlined in chapter 2) due to the recurrent nature which translates well into applications where knowledge of previous iterations is important.

The specific RNN algorithms used in this work are the long short-term network developed by Hochreiter and Schmidhuber (1997) as well as the gated recurrent units which were an evolution of the LSTM developed by Cho et al. (2014)

Many considerations as to the specific models used have been made to ensure optimal performance. A standard RNN uses feedback from outputs to change weights on inputs and thus affect the inputs. In this sense, an RNN has an element of awareness of previous input events (short-term memory) (Hochreiter and Schmidhuber, 1997). This can be seen in the figure 3.1 below. The problem with traditional recurrent networks is that the errors which are back-propagated can lead to oscillating weights, or cause the learning to take significantly longer due to the errors tendency to blow up or vanish (Hochreiter and Schmidhuber, 1997). Some of these issues are solved by the introduction of the Long-Short Term Memory (LSTM) RNN proposed by Hochreiter and Schmidhuber (1997) which uses a gradient based algorithm to calculate errors based on larger number of epochs (over 1000). This then tends to minimise the effect of noise in the input sample, and also reduces the chance for oscillating weights to occur.

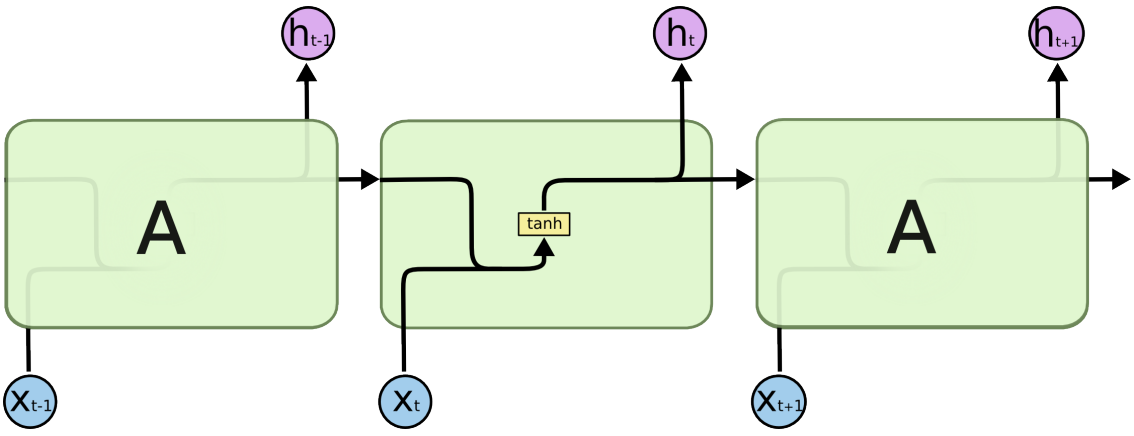

Figure 3.1: Diagram showing the structure of a standard recurrent neural network node (Olah, 2015)

3.2.1 Long Short-Term Memory (LSTM)

Fundamentally, an LSTM network consists of the cell state, carried from the previous layer to the next layer, which is adjusted based on a number of gates which decide how much of the data to let through. In a basic LSTM network there are 3 of these gates; the forget gate (which decides how much to throw away from the previous layer), the input gate, in combination with a *tanh* function, decides which values are to be updated, and the final output gate which affects how much the input is changed and also constrains the output to within a ± 1 range Olah (2015). These gates can be seen in figure 3.2 below with the forget, input and output gates shown from left to right respectively.

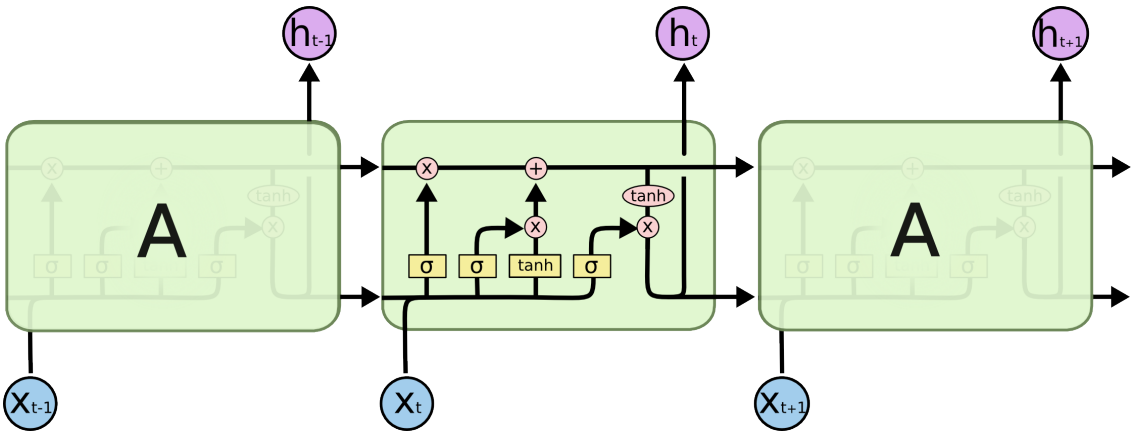

Figure 3.2: Diagram showing the structure of a standard LSTM network node (Olah, 2015)

3.2.2 Gated Recurrent Unit (GRU)

Jozefowicz et al. (2015) investigated whether the additional parameters used in an LSTM actually provided any improvements. They found in many cases that it was not necessary. A more specialised version of the LSTM network created to overcome this is the gated recurrent unit (GRU) (Cho et al., 2014) effectively combines the first two gates (input and forget) of an LSTM into an all-encompassing "update gate" Olah (2015). This alternative implementation was found to be more efficient in many language processing tasks (Chung et al., 2014) and is therefore the other type of recurrent network this report will investigate.

3.3 Implementation details

A model is constructed from GRU neurons and LSTM neurons alike, using the same setup where possible to confine the differences to the type of neurons used.

A gated recurrent unit neural network is implemented first to perform the learning and prediction task. The GRU model is built from neurons implemented in the Keras library[1]. The LSTM model is implemented the same way. The parameters used are manually specified to ensure optimum performance for training time as well as minimising the loss to achieve the best possible output. The specific parameters and functions used are justified below and shown in table 3.1.

[1] https://keras.io/api/layers/recurrent_layers/gru/ (accessed on 25 Mar. 2020)

3.3.1 Configuration of the network models

It is necessary to specify the parameters for the network including the input dimensions and the number of hidden RNN units in the neural network. It is also important to choose a loss function to decide after each epoch how closely the network represents the training data.

The parameters chosen are intentionally consistent between the GRU model and the LSTM model to ensure minimal differences which could hide the true strengths and weaknesses of each approach. The batch size is the number of training samples which are fed into the model per iteration. A batch size which is too low will therefore result in incredibly slow training. On the other hand, a batch size too large has been found to hinder the performance of the model in some cases (Keskar et al., 2016). A value of 64 was found to be a good balance which allowed training to pass at a reasonable speed.

The loss function used is a cross-entropy loss function, as outlined in the introduction and shown in equation 1.2. This is used to measure how closely the model aligns to the training data. A cross-entropy function is a form of negative log function used to calculate the differences between two probability distributions (Brownlee, 2019). When implemented, this results in a loss function which penalises the model more heavily the further the from the expected output it deviates. The same loss function is used for the GRU model and the LSTM model to allow more consistency and make the eventual comparison less subjective.

The optimizer used is the Adam optimizer, specifically the one implemented by the Keras library. The optimizer is the component in the model which adjusts the weights after each iteration to allow the loss to decrease. Adam is an adaptive optimizer, meaning a unique learning rate is used for each parameter, and the learning rates evolve over many iterations. This allows for significantly faster training. Adam also has momentum built into the optimization process, which helps to prevent the learning from being caught in small local minima.

Parameters	GRU	LSTM
Text Generator		
- Embedding dimension	256	256
- Batch size	64	64
- Optimizer	Adam	Adam
- Learning rate	0.01	0.01
- Momentum	0	0

Table 3.1: Configurations of network model Hyper-Parameters

3.3.2 Initialization of the training

At this point the model is ready to receive the training data that has been collated. Using the training sets that were created previously, the data is input in batches and training is initialised. The number of epochs that training is performed over must also be specified. Increasing this will affect the accuracy of the output so a few variations are tried.

3.4 Text Generation

Now that the neural network is trained, it can be tested. Feeding the model a string or a set of strings as a start point, it will generate a prediction of upcoming words based on the context of words used in the input sample. The output state of the network at each stage is

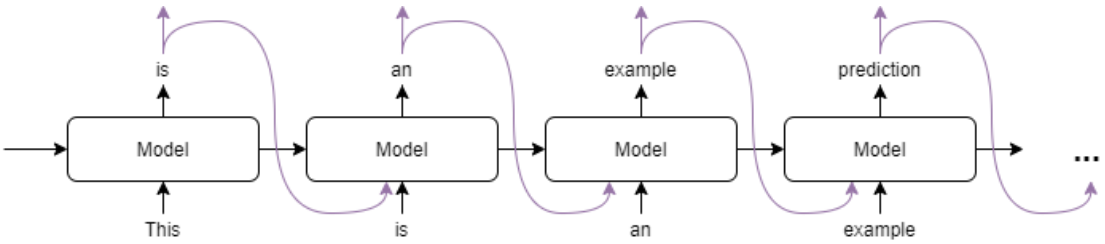

Figure 3.3: Flowchart representing the state of the model being carried forwards as input to the subsequent stages. Figure inspired by TensorFlow (2020)

then fed back into the network to generate the next values as shown in figure 3.3 above. This can continue until a string of the desired length has been created. Since the capitalization was removed whilst sanitizing the data, now is the opportunity to add it back, using a simple algorithm to capitalize the first words of each sentence.

3.5 Evaluating the generated text

Forming an evaluation of the text generated by the model is a crucial step in assessing the accuracy and ultimately the performance of the model. Sharma et al. (2017) implement an API to access a selection of language evaluation metrics which are used to quantitatively measure how well the model performs at generating text based upon a reference.

Using as reference input, all the tweets which share the same start string as the model used, and the final tweet as the string to evaluate, the metrics are calculated and compared. The metrics used include the BiLinugual Evaluation Understudy (BLEU) developed by Papineni et al. (2002) of which BLEU_1 - BLEU_4 are all used. It is important to use multiple of these BLEU metrics, particularly as the latter accommodates for extended n-grams in the string, which is a stronger sign of more consistent generated text. Since these metric were initially generated for language translation tools, the consistency of n-grams was a secondary concern, but is much more important in this application where text generation is the focus.

The remaining metrics, including METEOR and ROUGE amongst others, must also be accommodated for in a way that considers the strengths of each technique. METEOR, for example, will evaluate similarly to BLEU but will take into account synonyms used in the text. This makes it more forgiving in some cases. ROUGE on the other hand has its strengths in its ability to assess whilst considering the longest common sub-sequences (LCS) and overlapping sequences found in the text. This makes ROUGE a particularly strong consideration when evaluating a text generation tool.

3.6 Summary

The input data is gathered for an individual user on Twitter to be used to train the model. The tweets of @realDonaldTrump are used due to the large number of tweets available, which can help to ensure a more representative model is built. The text data is sanitized to ensure there are no invalid characters which would cause issues in the output text. The embedding process represents the text input as vectors which can be split into sequences to be used as the batch input for the networks. The embedding is done through an implementation of the continuous bag-of-words model.

The networks are built using the same hyper-parameters where possible to ensure the only variables are the neurons in the network. Cross-entropy is used as the loss function due to

its aggressive penalty for deviating from the expected output. Adam is the optimizer used, which adapts the learning rate for each of the parameters independently, helping to increase the momentum of the learning and avoid local minima.

Data is split into consecutive sequences which are grouped into batches to be fed into the network models. A string is provided to the model after training is complete to generate a tweet using the string as a seed. After the text has been generated from the model, it is evaluated using a selection of language evaluation metrics. Several metrics are used alongside each other to gather a more balanced view of the weaknesses and strengths of the text generation system.

Chapter 4

Results

The models were trained and results generated using a dataset of tweets from a specific user. The Tweets of @RealDonaldTrump were used for this example due to the large volume of tweets. A model constructed of Gated Recurrent Unit (GRU) neurons and a separate model of Long Short-Term Memory neurons were used to generate the tweets. The two different models learn in different ways and the results during the learning and the eventual tweets generated by each are highlighted in this section. The loss at each stage is calculated using a cross-entropy loss function, as described in chapter 3.

Figure 4.1: Example Tweets generated by a GRU and LSTM model respectively. Visuals created using TweetGen.com[1]

4.1 Gated Recurrent Unit (GRU) Model Text Generation and Evaluation

Using a network of GRUs, the model was trained on the dataset and tweets generated. The loss at each epoch of the training was initially very high (7.4258 at epoch 1 and 6.3071 at epoch 10). This was resolved over a large number of epochs. The process of training aims to reduce the loss at each stage, and this is shown in the fig 4.2 below.

After 200 epochs, the loss of the model is significantly lower than the starting loss at 0.3941. As the loss tends towards 0, the predictions become more and more accurate to the training data. The risk is that the model begins to "remember" the training data and will not

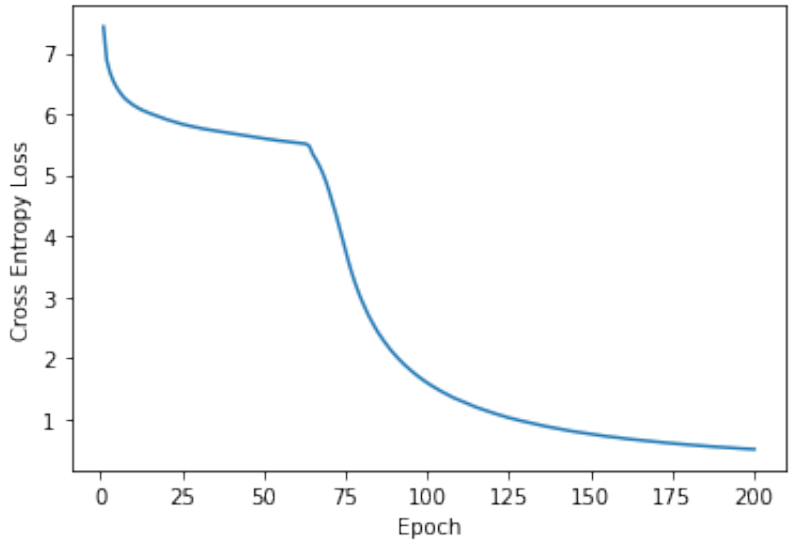

Figure 4.2: Graph showing the training loss over 200 epochs on a network of GRU neurons

work well for new test data. This is caused by over fitting. By stopping the training when the loss stops decreasing, we can find an optimal number of epochs.

4.1.1 Text Generated by Network of GRUs

The text is generated using a start string and the state of the model after each word is generated is used as the input for the next stage. Using a string as input to the model, an output tweet is generated. The quality of the tweets is also affected by the number of epochs the model is trained for. A example of the output from the model is provided in table 4.1 below.

Table 4.1: Tweets Generated from GRU Network Model

Start String	Epochs	Output String
barack obama	1	Barack obama @vincemcmahon #votetrump ! that. . I and in. . . . . In the very the good victims. . . . That called donald. To @sweettea2u for the the !. . I of it. . . . I. Of the be back
climate	1	Climate blockbusters pies usurp @gabrielc1997 regurgitation @betterhermann "unaffordable" wheeler anyone's krugman #white disgraces "more @svenisaksson #maitnerstrong @raffasolaries @tbclem2 @samertime hitter @thatbrody @trackdonny aya dipshit facilitating @mclaughlingroup @chef2234 air"thanks @jerrimoore "swamped" @joshhodgkinsonn @johndoe5210 @parisitony @mrtohney steam accumulation bestthx @reinosooo 50b @jeffferrier "mister @nishantagg23 @paula_white @lisa_goforth megrahi diplomatic @sassylassee travelling finalists @888maggie888 @spacemanchris

Continued on next page

Table 4.1 – continued from previous page

Start String	Epochs	Output String
barack obama	200	Barack Obama uses racial categories across congressman @davebratva7th is great fighter #crookedhillary. You can't be so lucky for our president but they can help! I believe in spending and we must have together next week by far lower premiums cost & deductibles than obamacare. Had it before
barack obama	200	Barack Obama fell 50 from a deserter bergdahl or repeal of repeal and replace of immigration with another iraq war or repeal or repeal and replace! rt @ivankatrump for those of those who make it proud of you! #trumprallydallas #kag2020 just arrived in new hampshire for a #maga rally in
barack obama	200	Barack Obama marco! Do we now get over but they are having a gun on its magic higher & high on the obamacare site. He will be forced for another season of star @celebapprentice. Despite the record number he in the white house christmas tree getting it.
climate	200	Climate dreams; advice for me as you can see; you are doing a terrific job! Just returned from mississippi great school with a fabulous world. " " @dagnyred @karentalk @foramerica @realdonaldtrump I'm latina legal & fully support you speaking the truth! Don't back down!
climate	200	Climate dreams; story to secure our border on a strong agreement. The world is watching and great less expensive mistake and go the chinese say! @foxnews that dopey graydon carter is about food and strength. Why can't he do it fix? Terrible five terms! "
climate	200	Climate dreams; advice as they go on stage you were meant to be destroyed 100 of their concentrated effort. Keep it up! via @newsmax_media by trump I give every american entrepreneurs can't lose by the reason via @newsmax_media "trump s rally for thursday 3 5 at 2
climate	200	Climate change policy in paris. Protests @chucktodd and @nbcnews enjoy the red line in charge of the ridiculous end of well! Thank you Brian and the democrats are not happy with you!!!!! This country needs you! #trump2016 #makeamericagreatagain ! All star celebrity @apprenticenbc
hillary clinton	200	Hillary clinton lover agent peter strzok texted his lover peter strzok texted his lover lisa page @judicialwatch a @judicialwatch @foxnews tony is not a long ago that he would be a star. If he and endless tweets weak advice so brave read the republican party see me tomorrow. Lets #makeamericagreatagain

Continued on next page

Table 4.1 – continued from previous page

Start String	Epochs	Output String
hillary clinton	200	Hillary clinton 60 get out! Why did senator from Florida and Florida with their partner in the end? rt @senatemajldr will be testify after nancy pelosi asking for herself as for one president when you are the president can get cont in California you can I see yourself!

4.1.2 Language Evaluation Metrics Applied to GRU Generated Text

Table 4.2: Evaluation Metrics for Text Generated from GRU Network Model

Text Evaluation Metric	Score after epochs	
	1 Epoch	200 Epochs
BLEU_1	0.03846153846079883	0.8823529411591696
BLEU_2	0.02746175181852122	0.6507913734430804
BLEU_3	2.4707501127559876e-07	0.37291762968874936
BLEU_4	7.448564757962441e-10	0.23860508271347589
METEOR	0.12477675437311822	0.1514624459011613
ROUGE_L	0.03935483870967742	0.23804878048780484
CIDEr	0.0	0.0
SkipThoughtsCosineSimilarity	0.5418774	0.51103306
EmbeddingAverageCosineSimilarity	0.706768	0.959474
VectorExtremaCosineSimilarity	0.368572	0.55877
GreedyMatchingScore	0.758379	0.721049

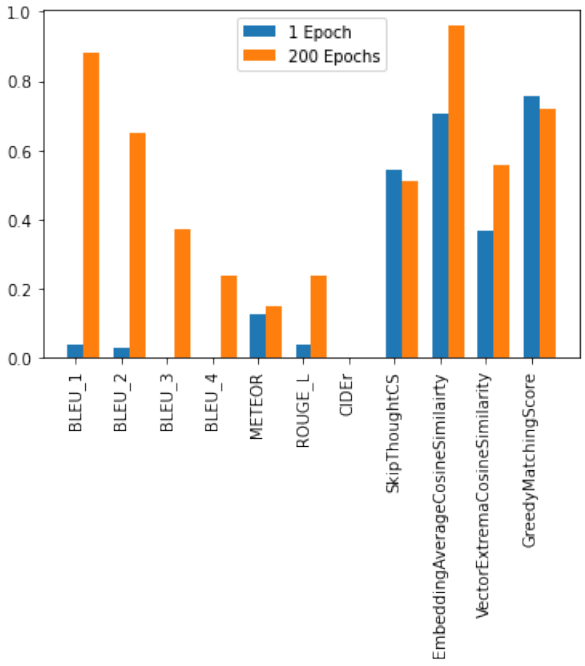

Figure 4.3: Graph showing the scores for various language metrics of the text generated using a GRU network

4.2 Long Short-Term Memory (LSTM) Model Text Generation and Evaluation

Using a similar approach, but training using a different type of neuron (LSTM) to make up the network, the results are shown below. The initial loss after a single epoch was 7.445, and after 200 epochs had reduced to 0.3466.

Plotting the loss against the epochs provides an insight into the progress of the learning and at what points the learning is completed. This is visible in the figure 4.4 below

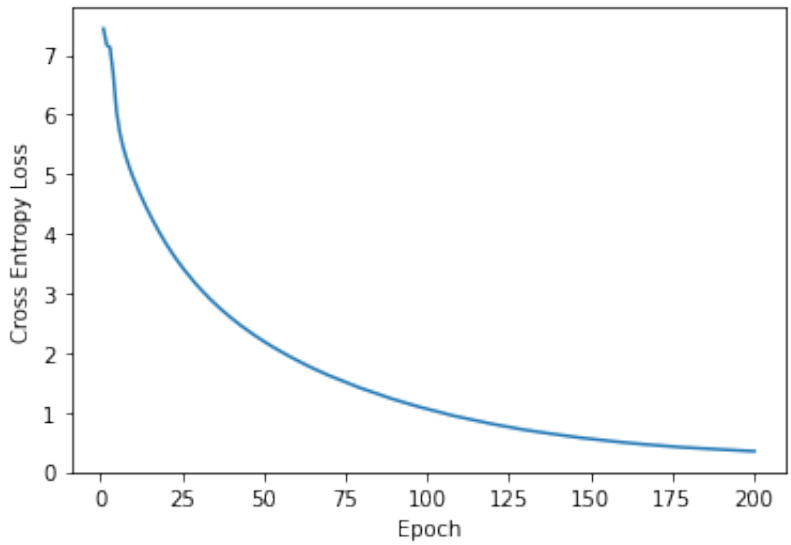

Figure 4.4: Graph showing the training loss over 200 epochs on a network of LSTM neurons

4.2.1 Text Generated by Network of LSTMs

An example of the outputs from this alternative network are once again shown below in table 4.3 alongside the initial input string and the number of epochs the network had been trained for.

Table 4.3: Tweets Generated from LSTM Network Model

Start String	Epochs	Output String
barack obama	1	Barack Obama reform @randyrolph underperforming damning... For to to. There. To are to donald. Of.. The " ! @realdonaldtrump and... And to.... And.? " a!. The 10 who a us what always the
barack obama	200	Barack Obama powerful " @rickclement @leezagibbons I agree thanks Donald!!!!!!!!!!!!!!!! " thanks! " @deadlykittenz @realdonaldtrump yay watching @cnn which is really right! Powerful rt @realdonaldtrump megan king who is running for superior court judge

Continued on next page

Table 4.3 – continued from previous page

Start String	Epochs	Output String
barack obama	200	barack obama should say to change that he is now selling the debt ceiling . " true. " @ellenespence @marklevinshow thank you for maintaining your integrity during this disgusting lynching of @realdonaldtrump" thanks mark!. @richlowry is truly one of the dumbest of the american middle class
barack obama	200	Barack obama so many amazing young entrepreneurs across America is becoming stronger and stronger together! Presidential harassment! The greatest overreach in the history of our country. The dems are obstructing justice and will provide whatever assistance is needed. We are ready willing and able. We are doing
climate	200	Climate @ro18007212 Missouri rt @realdonaldtrump congratulations to the 2016 @clemsonfb tigers! Full ceremony finally held our first full @cabinet meeting today. With this great team we can restore American prosperity and br congratulations! 'First new coal mine of trump era opens in Pennsylvania' we will never forget
climate	200	Climate and they were treated badly. Many of our difficulties. Thank you Iowa get out and vote! I am with you all the way !..... What? Did you ever see a situation room is totally wrong on him. " Arnold Palmer
climate	200	Climate @hannahbsampson rt if she should have been nicer and more respectful to his wife! He knows nothing about me or my religion. Came to my office looking for work. I never went bankrupt.... As I have been saying. African Americans will vote
hillary clinton	200	Hillary Clinton thank U.S.A. Wins! " @danielprofit @jskrepak @realjameswoods @realdonaldtrump you the better thing in my lifetime we are better than i can do now! " " @shea_leduc @realdonaldtrump for president! Watched @realdonaldtrump @ivankatrump on the five star premiere of @apprenticenbc the night show
hillary clinton	200	Hillary Clinton the white house judiciary committee? Stalling but for what reason? Not looking good! Lawmakers of the house judiciary committee are angrily accusing the department of justice of missing the thursday deadline for turning over unredacted documents relating to fisa abuse fbi comey lynch mccabe clinton emails and
hillary clinton	200	Hillary Clinton nonpolitical.. C. The wall is being built after victories against the democrats in various courts! rt @vp the #usmca is a deal for the 21st century that will support mutually beneficial trade and lead to freer and fairer markets! usmcan rt @vp the usmca is

4.2.2 Language Evaluation Metrics Applied to LSTM Generated Text

Table 4.4: Evaluation Metrics for Text Generated from LSTM Network Model

Text Evaluation Metric	Score after epochs	
	1 Epoch	200 Epochs
BLEU_1	0.846153846137574	0.9807692307503699
BLEU_2	0.6041585400074664	0.733799385691093
BLEU_3	0.30793846565631483	0.3776170216694533
BLEU_4	2.77842665099351e-05	0.18207052810731428
METEOR	0.1578992148171917	0.16274686398819752
ROUGE_L	0.3345161290322581	0.2558064516129032
CIDEr	0.0	0.0
SkipThoughtsCosineSimilarity	0.5624511	0.579224
EmbeddingAverageCosineSimilarity	0.968589	0.955188
VectorExtremaCosineSimilarity	0.577852	0.575335
GreedyMatchingScore	0.850724	0.75106

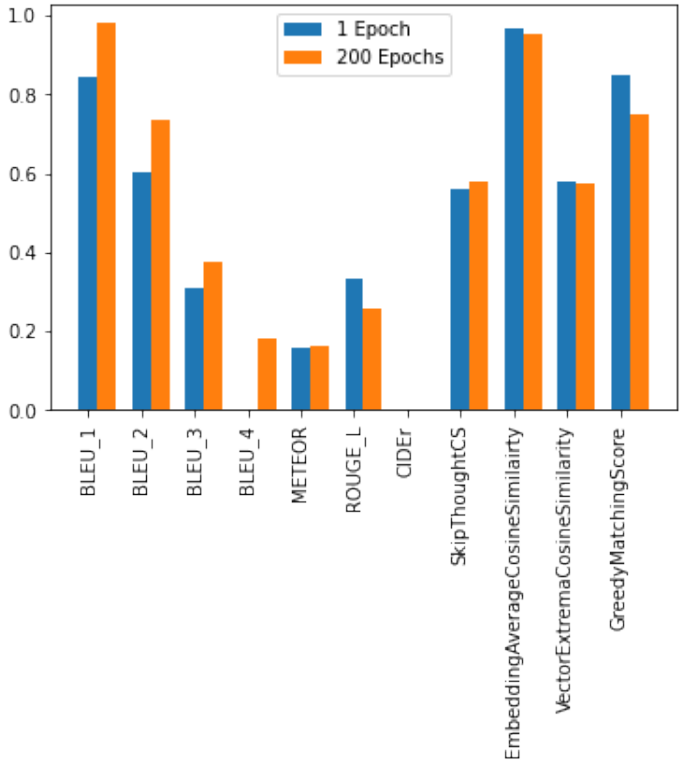

Figure 4.5: Graph showing the scores for various language metrics of the text generated using an LSTM network

4.3 Summary

The results show the tweets generated by each of the models over multiple iterations and using different start strings as the seed. Tweets generated become legible English sentences after

around 150 epochs. The cross-entropy loss over time decreases significantly with iterations. The loss of the GRU model shows a local minima found around epoch 70, which is overcome with further iterations. The loss of the LSTM model reduces faster than the GRU model and results in a lower loss after training is complete.

The tweets generated are evaluated against language metrics standards and the resulting scores plotted in a graph showing the improvements in the scores over many iterations.

Chapter 5

Discussion and Analysis

Both the GRU and the LSTM model were able to successfully generate text that begins to imitate the natural style of writing found in the the training data. The accuracy of this is partially assessed using a combination of the evaluation metrics.

A comparison of several metrics can be used discerningly to highlight the strengths of each approach and a combination provides insight into where a model excels or suffers.

5.1 GRU Tweet Generation

The tweets generated by the model built from GRUs begin to resemble legible English sentences after around 150 epochs. By considering the loss rate (as shown in figure 4.2), it is possible to see there was a local minima found at around 70 epochs, where the loss appears to stabilise. After further epochs however, the loss drastically reduces and begins to quickly tend towards 0.

5.2 LSTM Tweet Generation

Similarly to the GRU, increasing the number of epochs causes the loss to decrease, and therefore the model becomes a better fit of the input data. It tends towards 0 over many epochs. There is no appearance of any significant local minima in the training of the LSTM model unlike the GRU model, most likely caused by the combined gates present in the GRU.

5.3 Comparison of approaches

For both models, the number of iterations clearly has a significant impact on the performance. After only a single epoch, the results between the model built with GRU neurons and LSTM units perform almost identically. At this stage the model has not learned the data thus the output is largely random. The loss of the GRU model after a single epoch is high at 7.4258. At this stage, the text generation is largely random so this is to be expected. The LSTM model has an initial loss of 7.4451 which is also high but very close to that of the GRU, reinforcing the notion of a random generation, where at this stage, the difference between the models does not affect the results.

After further iterations, and as the models learn and adjust weights accordingly, the differences become more obvious. Using the same hyper-parameters for the GRU model and the LSTM model, the rate of learning of the LSTM model was initially significantly faster, shown by a steep gradient on the loss curve. After a few more epochs, it becomes apparent that the

learning rate is not consistent between the models. The GRU model initially appears to begin to tend towards a loss of around 5.5 but after further epochs, this is clearly a local minima. Beyond this, the learning rate picks up again, and this model too starts to tend towards 0.39.

After training for the same number of iterations, the models end in a loss very close to 0. The loss of the LSTM model after 200 epochs is approximately 0.35 suggesting a good learning of the training data. The GRU model ends with a loss of around 0.39, which while quite low, is not as low as the LSTM model, suggesting it is not able to model the input data quite as accurately. This could also be a sign of overfitting from the LSTM model but the difference between the loss on the two models is relatively small so this is unlikely.

One recurring theme in the output tweets which may stand out is the appearance of consecutive punctuation characters. This could be a flaw in the model, but is more likely representative of the input data so should not be taken as a strong indicator. This is confirmed by searching the input data and finding over 2000 occurrences of the string "....". From the text generation point of view, this would mean the probability of a '.' character given the previous character is '.' is relatively high.

Considerations of Language Metrics The loss of the model is not the only important factor when considering the results. Arguably more important is the output of the model, i.e. the tweets that are generated. When analysing the textual output of the model using a set of language evaluation standards, we can see the difference that higher epochs provide. The results after 1 epoch are expected to be more or less random so it is no surprise that the text evaluation metrics scored lower in the majority of cases for 1 epoch compared to 200. The use of the BLEU metric however (which scores based on the number of occurrences of n-grams in the sample data also found in the output), scores somewhat surprisingly well at low epochs. This is due to the linguistic context being ignored in BLEU and the score based solely on the vocab used. The BLEU_4 metric, which requires occurrences of longer n-grams, penalises the random nature of the early epochs more heavily and highlights the importance of considering the metrics used. BLEU_4 unsurprisingly scores incredibly low for the low epoch results (in the range of 1×10^{-10} for the GRU model and 1×10^{-5} for the LSTM model) but scores significantly higher after extended iterations. This is a good sign of consistent occurrences of longer n-grams found in the resultant tweets.

A careful consideration of all the language evaluation metrics shows a clear correlation between increasing the number of epochs and scoring more highly. This is what one would expect from a language generation model. The increased number of epochs results in lower loss of the model, and therefore the model becomes a closer representation of the training data. Assuming the same data from training is used in the evaluation, such as in this case, the higher score is the logical outcome.

5.4 Summary

The LSTM model performs incredibly well, with the learning rate being significantly higher than the GRU for a large portion of the training. This is somewhat surprising given the more complex nature of an LSTM unit. It is explained however by the additional parameters present in the hidden neurons of the LSTM unit, which in many cases, lead to improvements in accuracy, and therefore can increase the speed of training.

The text generated by the two models is similarly representative of the training data, as shown by the results of the language evaluation metrics. The LSTM model scores higher on average than the GRU model however, further supporting the LSTM as the superior model for this application.

Chapter 6

Conclusions and Future Work

This study shows that is certainly possible to generate tweets with reasonable accuracy based solely upon the tweets of an individual. If provided enough time and resources for training, the results can even be mistaken for genuine tweets.

6.1 Conclusions

Both the GRU model and the LSTM model were able to produce tweets after training on the same dataset, which performed significantly better than random generation. This is reinforced by the evaluation metrics which assessed the quality of the generated text, and found considerable improvements.

The use of either model would therefore be an acceptable choice for this application, but the LSTM outperforms the GRU slightly on several fronts. The LSTM model trained slightly faster than the model constructed of GRU's, and resulted in a marginly lower final loss of the model (0.35 as posed to 0.39 for the GRU). The speed of training has more practical advantages, especially where performance is a significant consideration. The lower loss also suggests a better reflection of the training data. To assert this as the case, the evaluation metrics were calculated and reaffirmed that the LSTM was able to achieve text generation which was more in line with the input data used as a reference.

The primary aim of this report was to investigate and evaluate the use of GRU and LSTM models to generate text based on learning from an individual users tweets. To that extent, this study has found conclusively that tweet generation for an individual is certainly possible, and GRU and LSTM models are acceptable techniques to achieve it. An LSTM model would be the preferred technique for the reasons outlines above, though with optimising of the parameters, the GRU is able to come incredibly close.

In the process of coming to this conclusion, the objectives as explored in chapter 1 were investigated. A network was built which utilised the gated recurrent unit, as well as a network which utilised the long short-term memory algorithm. They were trained on data from an individual users tweets and generated text in the form of tweets. The tweets generated were assessed using an array of language evaluation standards which highlighted the shortcomings and strengths which have been explored here.

6.2 Future Work

This study has highlighted that it is possible for smaller datasets to be used for text generation tasks, and it therefore opens up further opportunities in the field. This advancement would make it viable to set aside some of the training data to be used as a validation set, rather than relying on the same data for training and validation. This would enable a less biased evaluation of the results without suppressing potential over-fitting in the model.

The particular user that was chosen for this study (@realDonaldTrump) was chosen, primarily due to the large quantity of tweets and therefore to maximise the size of the dataset. Since we have seen that the size of the input does not have an adversely damaging effect on the results, it would be plausible to investigate the same process on other users with fewer tweets. If the results are able to come close to the performance measured in this study, this would have significant implications for the requirements for further research into text generation.

Chapter 7

Reflection

Through the process of conducting this piece of work, there have been several challenges faced and overcome beyond the technical challenges which are standard with a experiment this substantial. The requirement for advanced project management would be one of the more significant new processes that had to be undertaken. Ensuring that the report is updated regularly to prevent it becoming stagnant whilst balancing with other studies has been a challenge. Despite this, it is a vitally important process, and something which I am sure will carry across to many future applications.

The extended duration that this project spanned across, and the size of the project overall have required a new approach to ensuring quality is maintained. Whilst this does come to an extent with the balanced project management, it has also been a challenge in itself. The regular supervisor meetings and discussions have been an indispensable procedure to ensure quality did not slip between chapters and subjects.

The final, an perhaps most significant new concept to grasp has been the extensive independent study into a field I was largely new to. It has provided a necessary insight into the process of learning from academic literature and utilising multiple resources to come to independent conclusions.

Given the opportunity to face these challenges again, I would ensure the main aims and objectives of the project were cemented as early as possible to help to focus the research and investigations. Despite this, I feel the background knowledge of the field of natural language processing which I have picked up during preliminary investigations has inadvertently helped to create a more cohesive report.

Regardless, I do believe the final work produced is representative of the original aims, with a slight drift in focus towards a comparison of a few techniques rather than a deep dive into one technique in particular. It would be an interesting area for further study to investigate other techniques which can achieve the same or similar results.

References

Alvarado, J., Chen, C., Shirvani, R., Shkel, Y. and Williams, T. (2018), 'Identification and analysis of conversational codeswitching triggers'.

Botha, J. A., Pitler, E., Ma, J., Bakalov, A., Salcianu, A., Weiss, D., McDonald, R. and Petrov, S. (2017), Natural language processing with small feed-forward networks, *in* 'Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, Copenhagen, Denmark, pp. 2879–2885.
URL: *https://www.aclweb.org/anthology/D17-1309*

Bowman, S. R., Angeli, G., Potts, C. and Manning, C. D. (2015), A large annotated corpus for learning natural language inference, *in* 'Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)', Association for Computational Linguistics.

Brownlee, J. (2017), 'What are word embeddings for text?'. Accessed on 27 Apr 2020.
URL: *https://machinelearningmastery.com/what-are-word-embeddings/*

Brownlee, J. (2019), 'A gentle introduction to cross-entropy for machine learning'. Accessed on 27 Apr 2020.
URL: *https://machinelearningmastery.com/cross-entropy-for-machine-learning/*

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', *arXiv e-prints* p. arXiv:1406.1078.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014), 'Empirical evaluation of gated recurrent neural networks on sequence modeling', *arXiv preprint arXiv:1412.3555* .

Denkowski, M. and Lavie, A. (2014), Meteor universal: Language specific translation evaluation for any target language, *in* 'Proceedings of the ninth workshop on statistical machine translation', pp. 376–380.

Feng, W., Guan, N., Li, Y., Zhang, X. and Luo, Z. (2017), Audio visual speech recognition with multimodal recurrent neural networks, pp. 681–688.

Goldberg, E., Driedger, N. and Kittredge, R. I. (1994), 'Using natural-language processing to produce weather forecasts', *IEEE Expert* **9**(2), 45–53.

Goldberg, Y. (2016), 'A primer on neural network models for natural language processing', *Journal of Artificial Intelligence Research* **57**, 345–420.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. and Schmidhuber, J. (2016), 'Lstm: A search space odyssey', *IEEE transactions on neural networks and learning systems* **28**(10), 2222–2232.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long Short-term Memory', *Neural computation* **9**, 1735–80.

Jozefowicz, R., Zaremba, W. and Sutskever, I. (2015), An empirical exploration of recurrent network architectures, *in* 'International conference on machine learning', pp. 2342–2350.

Karani, D. (2018), 'Introduction to Word Embedding and Word2Vec'. Accessed on 3 Feb 2020.
URL: *https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa?gi=ae671688bb16*

Katte, A. (2018), 'This 2013 Seminal Paper By Google Researchers Changed NLP Forever. Here's A Deep Dive'. Accessed on 29 Nov 2019.
URL: *https://analyticsindiamag.com/seminal-paper-google-researchers-nlp/*

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. and Tang, P. T. P. (2016), 'On large-batch training for deep learning: Generalization gap and sharp minima', *arXiv preprint arXiv:1609.04836* .

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Lin, C.-Y. (2004), ROUGE: A package for automatic evaluation of summaries, *in* 'Text Summarization Branches Out', Association for Computational Linguistics, Barcelona, Spain, pp. 74–81.
URL: *https://www.aclweb.org/anthology/W04-1013*

Lin, C.-Y. and Och, F. (2004), Looking for a few good metrics: Rouge and its evaluation, *in* 'Ntcir Workshop'.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013), 'Efficient Estimation of Word Representations in Vector Space', *arXiv e-prints* p. arXiv:1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J. and Khudanpur, S. (2010), Recurrent neural network based language model, *in* 'Eleventh annual conference of the international speech communication association'. Accessed on 20 Apr 2020.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013), 'Distributed Representations of Words and Phrases and their Compositionality', *arXiv e-prints* p. arXiv:1310.4546.

Natural Language Processing: Definition of Natural Language Processing by Lexico (2020). Accessed on 21 Nov 2019.
URL: *https://www.lexico.com/en/definition/natural_language_processing*

NSS (2017), 'Understanding word embeddings: From word2vec to count vectors'. Accessed on 27 Apr 2020.
URL: *https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/*

Ojha, V. (2020), 'Artificial Intelligence Lecture - 8/10: Natural Language Processing - CS3AI18'.

Olah, C. (2015), 'Understanding LSTM networks'. Accessed on 10 Dec. 2019.
URL: *https://colah.github.io/posts/2015-08-Understanding-LSTMs/*

Pang, B., Lee, L. and Vaithyanathan, S. (2002), 'Thumbs up? Sentiment Classification using Machine Learning Techniques', *arXiv e-prints* p. cs/0205070.

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', Association for Computational Linguistics, pp. 311–318.

Reiter, E. and Dale, R. (1997), 'Building applied natural language generation systems', *Natural Language Engineering* **3**(1), 57–87.

Rezaie, A. (2015), Quasi-real time prediction of storm surge inundation for the coastal region of Bangladesh, PhD thesis.

Schmidhuber, J. (2014), 'Deep learning in neural networks: An overview', *CoRR* **abs/1404.7828**.
URL: *http://arxiv.org/abs/1404.7828*

Schmidhuber, J. (2017), 'Recurrent neural networks'. Accessed on 20 Apr 2020.
URL: *http://people.idsia.ch/ juergen/rnn.html*

Sharma, S., El Asri, L., Schulz, H. and Zumer, J. (2017), 'Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation', *CoRR* **abs/1706.09799**.
URL: *http://arxiv.org/abs/1706.09799*

Springer, S., Buta, P. and Wolf, T. C. (1991), Automatic letter composition for customer service., *in* 'IAAI', pp. 67–83.

Subramanian, S., Mudumba, S. R., Sordoni, A., Trischler, A., Courville, A. C. and Pal, C. (2018), Towards text generation with adversarially learned neural outlines, *in* 'Advances in Neural Information Processing Systems', pp. 7551–7563.

TensorFlow (2020), 'Text generation with an RNN'. Accessed on 14 Apr 2020.
URL: *https://www.tensorflow.org/tutorials/text/text_generation*

Twitter Developer API (2020).
URL: *https://developer.twitter.com/*

Voorhees, E. M. (1999), Natural language processing and information retrieval, *in* 'International summer school on information extraction', Springer, pp. 32–48.

Wakade, S., Shekar, C., Liszka, K. J. and Chan, C.-C. (2012), Text mining for sentiment analysis of Twitter data, *in* 'Proceedings of the International Conference on Information and Knowledge Engineering (IKE)', The Steering Committee of The World Congress in Computer Science, Computer . . . , p. 1.

Xie, Z. (2017), 'Neural text generation: A practical guide', *arXiv preprint arXiv:1711.09534* .

Zhang, X., Zhao, J. and LeCun, Y. (2015), Character-level convolutional networks for text classification, *in* 'Advances in neural information processing systems', pp. 649–657.