Neural Sequence Modelling for Automated Essay Scoring

Ahmed Hasan Zaidi
St. Edmund's College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: ahz22@cl.cam.ac.uk

June 10, 2016

Declaration

I Ahmed Hasan Zaidi of St. Edmund's College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 11,903

Signed:

Date:

Abstract

Automated Essay Scoring (AES) is the use of specialised computer software to assign scores for essays written in an academic environment. Its growing interest has been motivated by several factors including rising costs of education, need for grading standards, and major technological breakthroughs. Despite the positive results in literature, there still remain many critical challenges that need to be addressed to ensure the wide-spread adoption of AES systems. These challenges can be divided into three main categories: meaningfulness, transparency, and robustness.

This investigation aims to address these challenges while also attempting to improve the human-machine inter-rater agreement. Motivated by the recent success of neural networks, we conduct a systematic investigation of deep representation learning; initially using a basic recurrent neural network (RNN) but extending to Long-short term memory cells and deep bi-directional architectures as well. In order to evaluate the AES system, an adapted visualisation technique was implemented. The visualisation identifies portions of the text that are discriminative of writing quality.

Overall it was found that deep bi-directional model, DBLSTM, are more effective in capturing features discriminative of writing quality than shallower uni-directional models. Although the results did not surpass existing state-of-the-art, our methodology lays the foundations for a potentially rewarding avenue for future AES systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Automated Essay Scoring (AES) is the use of specialised computer software to assign scores for essays written in an academic environment. Its growing interest has been motivated by several factors including rising costs of education, need for grading standards, and major technological breakthroughs. With the recent advancements in machine learning and data science, there is an increased expectation in using technology to reduce the high costs of quality education. Currently, in order to ensure consistency across standardised examinations, academic boards hire multiple graders to moderate the scores on written assessments. This proves both time consuming and costly. AES proposes a system that can eliminate the need for additional markers but still ensure the necessary consistency required for comparable and reliable scores. The benefits of AES extend beyond just economic gains. Making the system available to less developed regions at negligible cost aligns with social responsibility of more developed countries in democratising education throughout the world.

Literature cites a variety of methods that have been used to develop AES systems. Larkey (1998) models AES as a text classification task and obtains high results by implementing a Naive Bayes model trained on vectors of stemmed words. Attali & Burstein (2006) attempt to use regression to build a generic scoring model, while Yannakoudakis et al. (2011) use an SVM and a preference ranking function to build a state-of-the-art AES system.

AES systems are evaluated according to inter-rater agreement between the human marker and the machine. Existing systems have demonstrated high levels of inter-rater agreement. However, despite the positive results, there still remain many critical challenges that need to be addressed to ensure the wide-spread adoption of AES systems. These challenges can be divided into three main categories: **meaningfulness, transparency, and robustness**.

In order to ensure that an AES system is **meaningful**, not only must the system must return high inter-rater agreement but the features being used in the model should also be truly discriminative of writing quality. This is to ensure the model encourages effective writing practices and not effective test taking practices. The second limitation of existing models is **transparency**. There is much skepticism surrounding the idea of relying on a mark that has been outputted from a black-box AES system. With existing models, assessors may not be able to isolate what areas of text were deterministic of the high/low grade. The third challenge of existing systems is **robustness**, particularly in their ability to identify and correct errors in the essays. By being able to identify and correct errors, the student and assessor can receive details on where the grammatical mistakes are, as well how they can be corrected (similar to human essay marking). Currently, only about 30% of errors made in written text are detected using state-of-the-art error detection systems (Ng et al. (2014)). A large number of missed errors are long-distance errors e.g. word order and agreement. In order for the AES system to make a valued and fair assessment of the essay, a larger portion of the errors need to be captured.

Motivated by recent advancements in deep learning, this investigation aims to address the three challenges mentioned above, by using recurrent neural networks (RNNs) to model the AES system. RNNs automatically abstract the features that are most discriminative of the defined problem (writing quality in the case of this investigation). The method of feature learning used by RNNs encourages better writing practices and prevents feature manipulation and reverse engineering thereby contributing to the meaningfulness of the model. In order to ensure transparency, a visualisation can be developed. This can identify the portions of the text that are discriminative of writing quality. Additionally, it can also be used in effective error analysis of the deep neural network.

The architecture of an RNN allows it to take inputs of variable size, making it ideal for processing essays with variable length. This along with the ability to emulate memory through the Long-short term memory (LSTM) cell provide RNNs with some promising scope in developing a robust AES system.

1.2 Purpose of Investigation

The purpose of this dissertation is to conduct a systematic investigation of deep neural networks on assessing upper-intermediate texts. More specifically, the aims of this study are to:

1. Carry out comprehensive literature review on existing techniques of AES and their limitations
2. Develop and extend an AES system using RNNs and its variations
    (a) Perform analysis on RNN and it's variants (including RNN-MLP, LSTM, BLSTM, DBLSTM) in determining writing quality and score
    (b) Identify the optimal hyperparameters that maximise the performance of the AES system
3. Exploit visualisation techniques to evaluate and conduct an in-depth error analysis of the AES system
4. Lay foundations for future research in the area of deep learning for AES

1.3 Contributions

The main contributions the investigation makes to the domain of Automated Essay Scoring are as follows:

1. A systematic investigation of AES systems using RNNs and four variations on assessing upper-intermediate texts:
    (a) Simple Recurrent Neural Network (RNN)
    (b) Recurrent Neural Network with Multilayer Perceptron overlay (RNN-MLP)
    (c) Long-short term Memory (LSTM)
    (d) Bi-Directional LSTM (BLSTM)
    (e) Deep Bi-Directional LSTM (DBLSTM)
2. A unique moethod visual error analysis of deep neural networks to improve transparency of the"black-box" model

1.4 Report Structure

This dissertation takes on the following structure:

1. **Chapter 2** provides a comprehensive literature review of previous AES systems and their limitations; a brief overview of natural language applications that make use of deep neural networks; and justification behind *some* decisions made during the development of the RNN models.
2. **Chapter 3** provides the background information and intuition behind the RNN and the LSTM. Additionally, the chapter provides some insight into an optimisation technique known as RMSProp.
3. **Chapter 4** presents an overview of the design and implementation of the five types of RNN models; the motivation behind the methods and design decisions is also discussed in this chapter.

4. **Chapter 5** provides a quick overview of the experimental setup, results of three different experiments (hidden-layer, script type, and learning rate) and a detailed analysis of the models in general.

5. **Chapter 6** describes the visualisation developed in order to understand the inner workings of the system and provide a valuable error analysis to further improve the existing models. This chapter also evaluates the meaningfulness, transparency, and robustness of the models..

6. **Chapter 7** draws general conclusions regrading the investigation and measures the contributions made against the key aims set out at the beginning of the study. This chapter also provides ares for future research.

Chapter 2

Literature Review

2.1 Automated Essay Scoring

There is a significant amount of research done in the area of AES. Project Essay Grade (PEG) developed by Page (1968) and Page (1967) uses manually identified textual features that represent proxies for writing ability. Some of the features include essay length, number of pronouns, POS tags, number of punctuation marks, presence of a title and number of paragraphs. The learning model then learns the essay scores by applying linear regression to the feature combinations. However, the features engineered by Page (1968) and Page (1967) such as essay length and number of paragraphs are easy to manipulate and does not encourage improved writing competence.

eRater by Attali & Burstein (2006) was the first AES system deployed in a high-stakes assessment. The system employs vectors and weighted features. The features are divided into several categories including grammar, style, mechanics, organisation and discourse, and semantic coherence and similarity. The model makes an assumption that well written essays are similar to other well written essays. Therefore, if the quality of an essay can be represented by a vector, the quality of other essays can be determined by taking the cosine similarity of the well written essay and the unknown essay. Linear regression was used to fit the model into predetermined marking schemes. Although Attali & Burstein (2006) presents a reasoned argument for vector representations, the features contained within the vectors are limited when evaluated for robustness. Furthermore, some of the manually engineered features require re-training and re-optimisation when used with a different dataset.

Larkey (1998) was one of the first to approach AES as a classification task by using Naive Bayes trained on vectors of stemmed words. This trend was continued by Rudner & Liang (2002) who developed Bayesian Essay Test Scoring sYstem (BETSY). BETSY uses Bernoulli Naive Bayes to classify text. The features include bi-gram, essay length, number of verbs, and noun-verb pairs. The models, however, do not make use of all of the training data available but rather train a separate classifier for each grade boundary.

Chen et al. (2010) use supervised clustering and a voting algorithm to score essays. Each of the texts are clustered according to grade and according to their respective z-score. At each iteration of the model, the score of the text changes based on similarity to all of the other texts until it reaches below a certain threshold. The model uses a bag-of-words representation. However, this makes it easy to reverse engineer the system and can encourage good test taking strategies as opposed to good writing techniques.

Briscoe et al. (2010) approached AES from a discriminative model approach. The motivation behind this decision was grounded in the assumption that generative models cannot make the "correct" assumptions regarding the quality of writing. As discriminative models making weaker assumptions, Briscoe et al. (2010) developed an AES model based on a variant of the batch perceptron algorithm. The model is trained on the Cambridge Learner Corpus (CLC) and employs lexical and grammatical features (e.g. POS tags, ngrams).

Inspired by the work of Briscoe et al. (2010) in using discriminative models, Yannakoudakis et al. (2011) developed a novel AES model using a learning-to-rank algorithm and SVM. Using a range of manually engineered features including lexical ngrams, POS ngrams, feature representing syntax, script length and error rate. Although achieving state-of-the-art performance and integrating the inherent “rank” structure of essays in the model, the features used in the model are prone to manipulation. Script length features may demonstrate high levels of correlation between the score, but is not necessarily indicative of good writing quality. Futhermore, error rate detection has not yet reached level of accuracy that justifies its use in a feature vector.

McNamara et al. (2015) approaches AES through a hierarchical classification model. It employs linguistic, semantic, and rhetorical features. Although perform relatively well on the given dataset, the model does not generalise well across different datasets, as it requires substantial modification to the feature engineering when applied to different types of texts.

Alternative approaches to AES have made use of a range of different features. Klebanov & Flor (2013) developed a new representation of content that captures levels of highly associated, mildly associated, unassociated, and dis-associated pairs of words. The study presents a relationship between word association and writing quality to evaluate essays written by college graduates. It was shown that high scoring essays contained higher proportions of highly associated and dis-associated pairs. Somasundaran et al. (2014) exploits local lexical chains to measure the discourse coherence quality in essays. The study shows that combining lexical chaining features with complementary discourse features can yield convincing results.

2.2 Discussion

There is strong evidence to suggest that the initial criteria set for an AES system that includes **transparency**, **meaningfulness**, and **robustness**, was not fulfilled by the aforementioned methods in literature. Thus far, experts in this field have relied heavily on feature engineering to obtain reliable results. However, those models do not generalise well across different types of essays. One can also call into the question the authenticity or meaningfulness of the proxy features implemented in the models. A specific example from the current state-of-the-art (Yannakoudakis et al. (2011)) includes script length. Many exams require their intermediate or advanced students to write longer essays while their beginner level students are tasked with a shorter length response. This can lead to the model learning features that do not reflect writing quality.

The recent breakthroughs made in deep learning, specifically in the domain of natural language processing, pose an interesting avenue of exploration for the future architectures of AES systems. Mikolov, Kombrink, Deoras, Burget & Cernocky (2011) and Chelba et al. (2013) have shown how RNNs can effectively be used for language modelling. These same language models can form the foundations of the approaches to other text oriented tasks. Motivated by the novel implementations of LSTMs by Tang (2015) and Tai et al. (2015), as part of this investigation we aim to explore a range of sequential neural networks, from the most basic RNNs to complex deep bi-directional LSTMs.

Chapter 3

Background

This chapter will present the relevant background material required to thoroughly understand the design and implementation phase of this investigation. Additionally, it will aim to provide some intuition regarding the design decisions made throughout this study.

3.1 Neural Word Embeddings

"You shall know a word by the company it keeps" - J.R. Firth (1957)

Inspired by one of the most successful ideas in statistical natural language processing, the AES system needs to incorporate a method of capturing the semantics of a word by observing the context in which it occurs. This can be achieved by using word embeddings (Bengio et al. (2006)). A word embedding is a vector of a predefined size, that when trained, aims to capture a distributional numerical representation of the word features. How well the vectors are trained depends on the method, as well as the quality and volume of data used.

Neural distributional representation is a popular technique of training word embeddings. It has been cited extensively throughout literature: Mnih & Hinton (2007); Collobert & Weston (2008); Turian et al. (2010); Collobert et al. (2011); Mikolov, Kombrink, Burget, Černockỳ & Khudanpur (2011). The embeddings combine vector space semantics with the prediction of probabilistic models. Each word is represented as a **dense** vector. This is contrastive to the 1-hot vector method where each word vector is the size of the entire corpus vocabulary (scarce vector).

Mikolov, Kombrink, Burget, Černockỳ & Khudanpur (2011) developed Word2vec, a popular model that trains word embeddings. The authors' view words as discrete states for which they are trying to determine transitional probabilities. The aim of Word2vec is to group similar words together on a feature space thereby making the processing of deriving similarities between words mathematically feasible. The architecture proposed by Mikolov, Kombrink, Burget, Černockỳ & Khudanpur (2011) is a two-layer neural network that takes a text corpus as an input and outputs feature vectors; each vector represents a word in the corpus. Word2vec is comprised of two models, the skip-gram model Mikolov, Chen, Corrado & Dean (2013) and the continuous bag-of-words model (CBOW) Mikolov, Sutskever, Chen, Corrado & Dean (2013).

As seen in Figure 3.1, Word2vec trains words against other words that neighbour them in the corpus. This can be achieved by using the word to predict the context (skip-gram) or the context to predict a word (CBOW).

Neural word embeddings have shown to boost performance for various natural language processing tasks from syntactic parsing (Socher, Bauer, Manning & Ng (2013)) to sentiment analysis (Socher, Perelygin, Wu, Chuang, Manning, Ng & Potts (2013)). Motivated by its performance, for this investigation we will be using pre-trained Word2vec embeddings[1]. The word embeddings have been trained on the Google News dataset (100 billion words) containing 3 million word vectors with dimensions of 300.

[1]pre-trained vectors available here http://code.google.com/archive/p/word2vec/

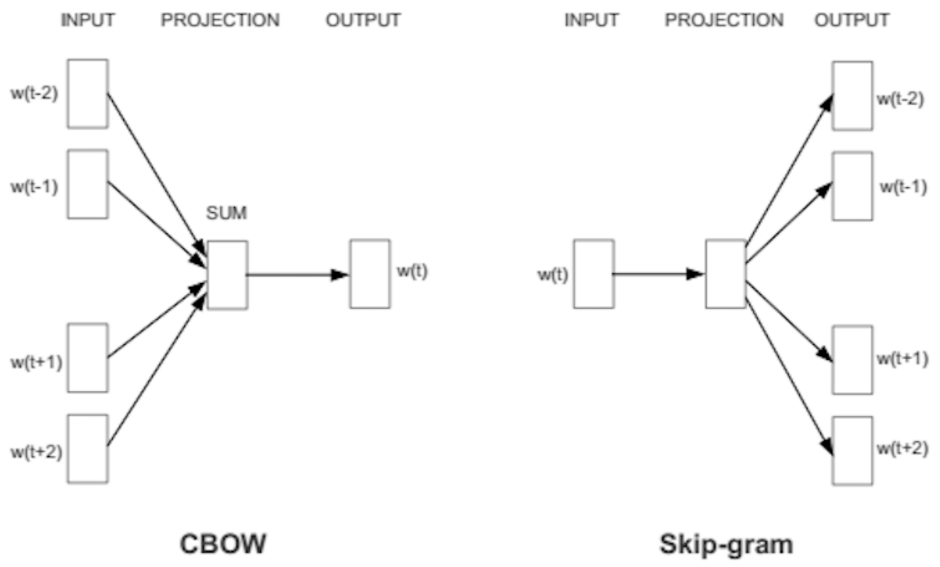

Figure 3.1: Figure showing Word2Vec architecture with both CBOW and Skip-gram methods of training

3.2 Recurrent Neural Network (RNN)

In order to understand RNNs, we must first revisit the basics of feedforward neural networks, specifically the multilayer perceptron (MLP). An MLP is made up on an input layer x_i, a hidden layer h_i and an output layer o_i. Formally, an MLP with a single hidden layer is a function f: $R^I \rightarrow R^O$ where I is the size of the input vector x_i and O is the size of the output vector o_i. The model also contains two weight matrices that must be optimised throughout the training process. The size of the matrices are determined by the size of each of the three layers (input, hidden, output).

The hidden layer h_i can be defined as follows:

$$h_i = s(x_i * W_x + b_x) \tag{3.1}$$

s is a non-linear function such as *sigmoid* or *tanh*. W_x is the weight parameter matrix for the input. It is the weight matrices that we are optimising through the model. b_x is the bias for the hidden layer. The hidden layer then feeds into the output layer o_i. The output layer can be defined as follows:

$$o_i = G(h_i * W_o + b_o) \tag{3.2}$$

G is also a non-linear function that can take the form of a softmax (in the case of multi-class classification). As we will be exploring regression as part of our experiment, we will not be implementing the function G. h_i is the output of the hidden layer which is multiplied by W_o, another weight parameter matrix.

In order to train the MLP, we use Stochastic Gradient Descent (SGD). The parameters of the model defined by $\theta = (W_x, W_o, b_x, b_o)$. For each parameter we calculate the gradient using the **backpropagation** algorithm (Rumelhart et al. (1988)).

For an MLP or feedforward neural networks in general, there is no notion of sequences or order in time. Particularly with essays, the sequence of words can provide substantial information regarding the essay as a whole. Fortunately, RNNs were built with the purpose to capture a series of events (training examples).

An RNN is a class of artificial neural networks that contain at least one feedback connection. At each iteration or time step t, the RNN takes an input x_t which is passed through the hidden layer s_t to the output o_t. The hidden layer of the previous time step s_{t-1} is also feeds into the s_t. This allows for temporal processing and sequence learning. Some refer to this feedback layer as a form of recent memory. In order to train RNN, **backpropagation through time** (BPTT) algorithm is used.

The RNN has proven to be successful across a variety of natural language processing tasks including statistical machine translation (Cho et al. (2014)) and language modelling (Bengio et al. (2006)) to name a few. However, its exploration in the domain of AES has been limited to the best of our knowledge.

The RNN network can be formalised as follows:

$$s_t = f_h(x_t, h_{t-1}) \tag{3.3}$$

$$o_t = f_o(h_t), \tag{3.4}$$

More specifically, each of the inputs, in equation 4.2 and equation 4.4 have associated weight matrices which are the parameters θ of the model. Therefore, the hidden layer s_t and the output layer o_t can be defined as follows:

$$s_t = f_h(Ux_t + Wh_{t-1}) \tag{3.5}$$

$$o_t = f_o(Vh_t), \tag{3.6}$$

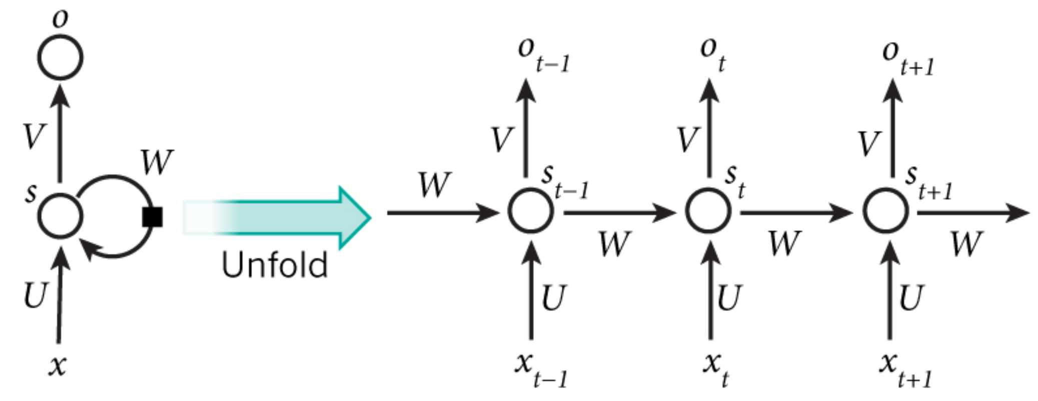

Figure 3.2: Figure showing a traditional RNN with one hidden layer (Source: Nature)

3.3 Long-short term memory (LSTM)

In a traditional RNN, the gradient signal is multiplied many times. This can be problematic for the RNN. More specifically, if the lead eigenvalues of the weight matrix is smaller than 1, the can results in the **vanishing gradient**

descent problem. Similarly, if the leading eigenvalue is greater than 1, the RNN will witness what is referred to as **exploding gradients**.

In order to address this problem, Hochreiter & Schmidhuber (1997) introduce Long-short term memory (LSTM). The LSTM is that uses "self-connected unbounded internal memory cells" (Graves et al. (2004)) that ensure a constant error flow. This allows the model to capture information across a wide range of timescales. LSTMs have been implemented effectively across many natural language processing tasks, including speech recognition (Graves et al. (2004)), context free grammar (Gers & Schmidhuber (2001)) and generating music (Eck & Schmidhuber (2002)); all tasks that place importance on the sequence of events.

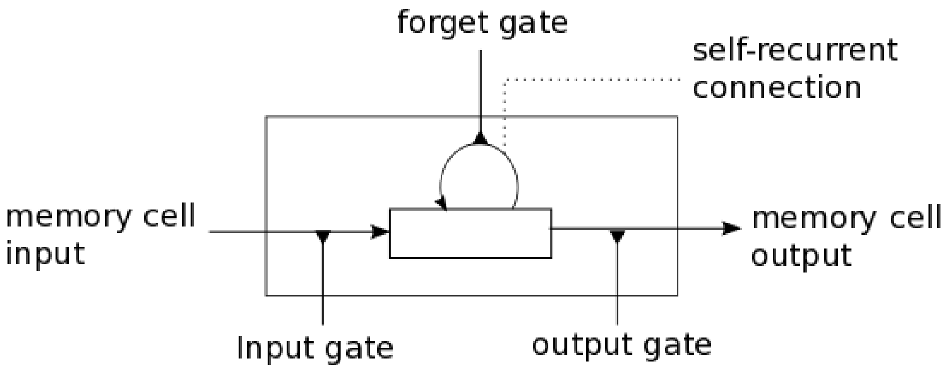

Figure 3.3: Figure showing long-short term memory cell with forget gate (Source: deeplearning.net)

LSTMs can be formalised as follows where i_t represents the **input gate**, $\hat{C}_t$ is the **current cell state**, f_t is the **forget gate**, C_t is the **new cell state**, o_t is the **output gate**, and h_t is the output of the LSTM. x_t is the input vector at time t. W_i, W_c, W_f, W_o, U_i, U_c, U_f, U_o, and V_o are weight matrices that are updated throughout the training process. The following parameters are bias vectors: b_i, b_c, b_f, b_o. The sigmoid function used as the non-linear activation function is represented by σ. The block in the middle of Figure 3.3 is often referred to as the **Constant Error Carousel (CEC)**. The LSTM model can be formalised as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.7}$$

$$\hat{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.8}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{3.9}$$

$$C_t = i_t \hat{C}_t + f_t C_{t-1} \tag{3.10}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \tag{3.11}$$

$$h_t = o_t \circ \; tanh(C_t) \tag{3.12}$$

3.4 Optimising the Model

3.4.1 RMSProp

RMSProp, initially introduced by Tieleman & Hinton (2012), is a method of iteratively decreasing the learning rate whilst preventing the issue of diminishing learning rates (an issue with Adagrad (Duchi et al. (2011))). RMSProp divides the learning rate for weight by the running average of the magnitudes from previous gradients for that particular weight. RMSProp can be formalised as follows where $E[g^2]_t$ is the running average of the magnitudes from previous gradients for that particular parameter, θ is the parameter being updated, and lr is the learning rate:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_2^t \tag{3.13}$$

$$\theta_{t-1} = \theta_t - \frac{lr}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{3.14}$$

Chapter 4

Approach

The literature review has revealed that the majority of the work done in AES relies on feature engineering. Motivated by this and the recent breakthroughs in deep learning, we decided to take on a new approach to AES with automatic feature recognition or **representation learning**. This chapter sets out to provide an overview of the approach used to develop an AES system using representation and deep learning methods. The chapter begins with explaining the data processing task, including providing an overview of the corpus. The second part of the chapter details technical aspects regarding each of the five models developed as part of this investigation along with design decision justifications.

4.1 Data Processing

This section describes the methodology of processing the data from the First Certificate in English (FCE) dataset. In order to input the essay into the deep learning models, several transformations must take place.

4.1.1 FCE Dataset

One of the datasets fundamental to improving the performance existing AES systems is the Cambridge Learner Corpus (CLC). The CLC is a collection of exam scripts from students learning English in collaboration with the Cambridge English Language Assessment. The dataset contains scripts from more than 180,000 students, from 200 countries and 138 different first languages. The word collection size is upwards of 50 million. The FCE dataset is a subset of the CLC. It contains a set of 1,244 exam scripts written by students sitting the Cambridge ESOL First Certificate in English. Each exam scripts includes the original text from the students, as well as annotations in the form of demographic details, error types, parts-of-speech tags (extracted from RASP parser).

4.1.2 Data Extraction

The FCE data is contained in three .xml files: training, development, and test. Each essay script in the .xml file is referenced by the tag `<script>`. We iterate through each essay in the file and extract the script level score referenced by `<g>` (range from 1-40) and the associated words referenced by `<word>`. Each essay also contains answer level scores (each script contains at least one answer) referenced as `<score>` (range from 1-20). Two separate datasets are created: 1) script level essays and scores; 2) answer level essays and score.

Dataset	Script	Answer
Training	1061	2116
Development	80	159
Test	97	194

Table 4.1: Table showing the number of essays in the training, development and test sets

4.1.3 Tokenising

After extracting the relevant data from the .xml files, the essay scripts must be tokenised. Tokenisation is the process of breaking up streams of text into individual words called *tokens*. Each token will receive a unique identification code or a key. This key will function as an index. Thereby, an essay can be expressed as a list of indices which refer to a specific word in the token dictionary.

For this investigation, we will only be tokenising words that occur at least twice in the entire dataset. Words occurring less than two times will be treated as unknown words. The justification behind this decision is based on the assumption that the usage of unknown words in an essay contribute more information about the quality of writing than the use of a particular word that has only been seen once. If unknown words are treated as a collective, we may be able to extract more information about the text.

4.1.4 Neural Word Embeddings

Each token or word in the token dictionary must be mapped on to a neural word embedding. This will allow each word to be "machine readable". We will use the pre-trained Word2vec representations to extract the neural word embeddings. A word embedding matrix is created containing a reference (index) to each word in the corpus and it's vector representation.

4.2 Model 1: Preliminary RNN

The first model used to develop the AES system was the vanilla RNN. As described in Section 3.2, the model contains a input layer x_i, a hidden layer h_i and an output layer o_i. The objective of the RNN, similar to all other machine learning algorithms, is to minimise the cost function. The cost function being used throughout this investigation is **least squares**, a standard approach to regression analysis. Least squares can be defined as follows:

$$cost = \sum (y_i - \hat{y_i})^2 \tag{4.1}$$

y_i is the target score for the essay while $\hat{y_i}$ is the model determined score for the essay. The objective of this model is to minimise the cost function, which in turn minimises the root mean squared error (RMSE) metric. Ideally, minimising the cost should improve the Pearson and Spearman assuming the model is not overfitted to the training data. The parameters that need to be optimised are outlined in Table 4.2

P	Dimensions	Description
U	input x hidden	weight matrix for the input to $hidden_t$
W	hidden x hidden	weight matrix for the $hidden_{t-1}$ to $hidden_t$
V	hidden x out	weight matrix for $hidden_{t-1}$ to output

Table 4.2: Table showing the dimensions of the weight matrices in an RNN

As mentioned in Section 3.2, the RNN is trained using BPTT. BPTT calculates the gradient of each cost function w.r.t the parameters through time. The cost function is then multiplied by a float known as the **learning rate**. The product of the learning rate and the gradient of the parameter is then subtracted from the previous value of the parameter. This iterative process known as **stochastic gradient descent (SGD)**, decreases the cost value of the model. However, the issue arises when selecting the right learning rate for the model. Choosing a learning rate that is too high results in the model not reaching the local or global (ideally) minimum as it may overshoot. Choosing a learning rate that is too low results in a model that takes a long time to converge (computationally expensive).

Ideally the learning rate should decrease while the model is being trained (i.e. get lower as the model gets closer to the local minimum). There are many methods of doing this, but one popular method of optimising the learning rate is RMSProp. The technical details regarding RMSProp are discussed in

Section 3.4. Please note that RMSProp is used throughout the investigation and is the preferred method of optimising SGD. RMSProp takes three main hyperparameters: learning rate lr, decay rho, epsilon ϵ. The values for decay and epsilon are set to 0.9 and 10^{-6} respectively, as suggested by Tieleman & Hinton (2012). The learning rate used for this investigation ranged from 0.001 to 0.01.

Another common issue with RNNs is that they are prone to overfitting. Overfitting is when the model fits a particular training set relatively well, but fails to generalise across the unseen test set. In order to prevent this we use L2 regularisation.

The RNN model outputs at every time step. More specifically, if an essay contains 300 words, during a single run-through of the entire essay, the model will output a score 300 times. However, for this investigation, we are only concerned with the score that is obtained in the final run i.e. after the entire essay as been processed. This is called the **many-to-one** approach (Please see Figure 4.1).

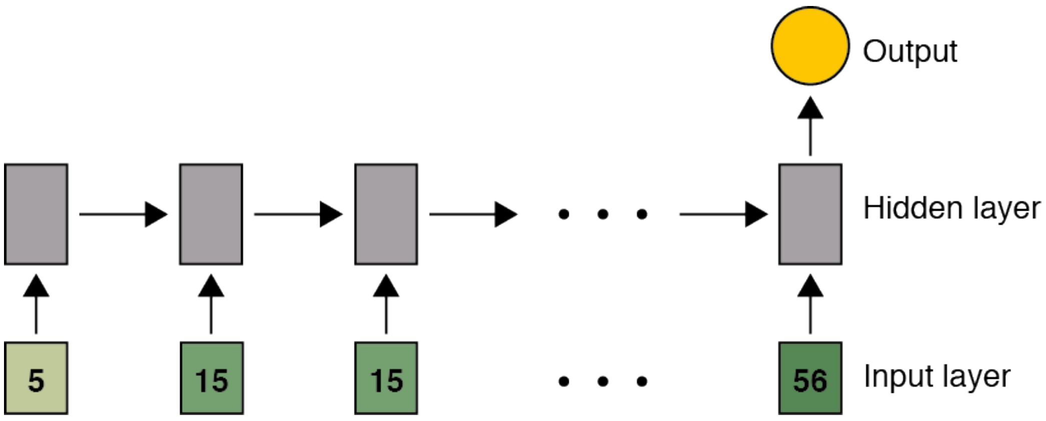

Figure 4.1: Figure showing the RNN with many-to-one output method

Table 4.3 outlines the hyperparameters that need to be tuned during the training process. The optimal hyperparameters for the RNN will be discussed in Chapter 5.

The model was trained for 50 epochs. An **epoch** is a single run through

Hyperparameter	Value
Learning Rate lr	0.001-0.01
Hidden Layer	50,100
L2-Regularisation α	0.001
Epsilon (RMSProp) ϵ	10^{-6}
Decay (RMSProp) rho	0.9

Table 4.3: Table showing the various parameters of an RNN

the entire training set. 50 will the be the default number of epochs for this investigation.

4.3 Model 2: RNN-MLP

The second model developed is the RNN-MLP model. This is a hybrid model that combines an RNN and an MLP overlay. The model functions similarly to the previous model described in Section 4.2. However, rather than having an output layer of with the size of 1, the model will output an embedding of size $50-100$ (depending on the configuration). This embedding will serve as the input to the MLP hidden layer which then outputs the essay score. The intuition behind using this model is that by applying an MLP overlay we can capture a higher level feature embedding for the entire essay. This allows us to emulate the complexity inherent in determine the quality of writing.

The RNN-MLP model can be formalised as follows:

$$s_t = tanh(Ux_t, Wh_{t-1} + b_s) \tag{4.2}$$

$$q_t = \sigma(Ps_t + b_q), \tag{4.3}$$

$$o_t = Vq_t + b_o, \tag{4.4}$$

The RNN-MLP model contains a few extra weight parameters in comparison to the RNN.

P	Dimensions	Weight Matrix Description
U	input x hidden-rnn	matrix for the input to $hidden_t$
W	hidden-rnn x hidden-rnn	matrix for the $hidden_{t-1}$ to $hidden_t$
P	out-rnn x hidden-mlp	matrix for the RNN *output* to MLP $hidden_t$
V	hidden-mlp x out-mlp	matrix for MLP $hidden_t$ to MLP output

Table 4.4: Table showing the dimensions of the weight parameters for an RNN-MLP

Hyperparameter	Value
Learning Rate lr	0.001-0.01
Hidden Layer (RNN)	50,100
Hidden Layer (MLP)	50,100
L2-Regularisation α	0.001
Epsilon (RMSProp) ϵ	10^{-6}
Decay (RMSProp) rho	0.9

Table 4.5: Table showing the various parameters for the RNN-MLP

Table 4.5 outlines the hyperparameters that need to be tuned. The optimal hyperparameters for the RNN-MLP will be discussed in Chapter 5.

4.4 Model 3: LSTM

As referenced in Section 3.2, the LSTM model was introduced in order to solve the issue of vanishing gradient descent prevalent in RNN models. LSTMs have shown to perform very well on tasks that require it to act based on information several time steps before (Hochreiter & Schmidhuber (1997)). However, as mentioned by Gers (2001), critical markers were processed along with the text to facilitate the model in learning when to use the crucial information. In many cases, it may not be possible to pass markers into the model. Gers (2001) develop an extension to the LSTM known as **"Peephole Connections"** that aims to allow the network to represent the duration of task-specific intervals without needing markers.

This modification is particularly useful in the case of AES as many of the

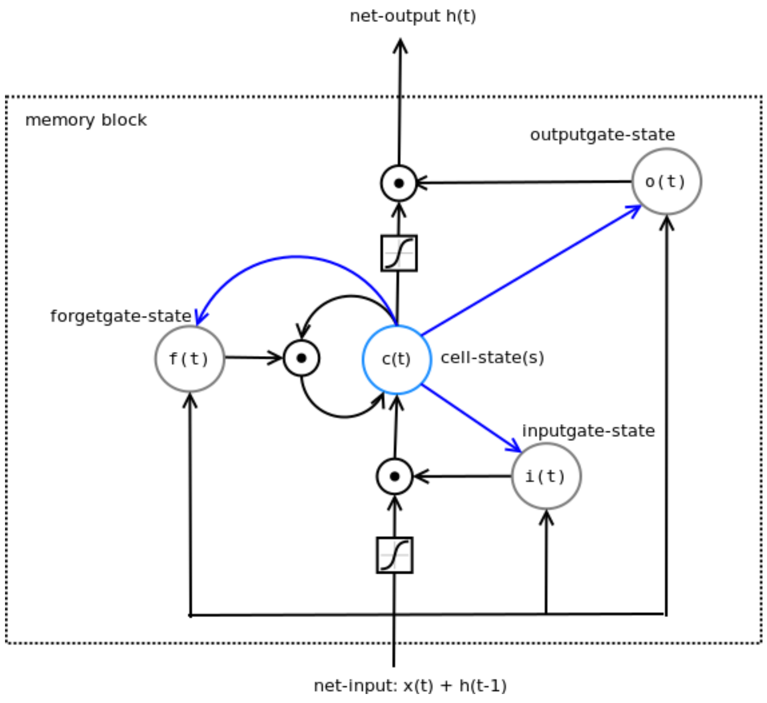

Figure 4.2: Figure showing an LSTM with "Peephole Connections" (Source: Herta (2015))

features inherent to writing quality are dependant on the precise duration of intervals between events. Suppose the network is trying to identify whether an essay contains periods in the right location. The network needs to learn the representation sentence and when the sentence is complete, if there is a valid punctuation present, the LSTM needs to know whether to allow the internal state to affect the output. In order to do this, the output gate must be aware of the contents in the internal state. The LSTM model with the "Peephole Connection" modifications can be represented by the following equations:

$$\mathbf{i_t} = \sigma(W_i x_t + U_i h_{t-1} + P_i C_{t-1} + b_i) \tag{4.5}$$

$$\mathbf{\hat{C}_t} = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{4.6}$$

$$\mathbf{f_t} = \sigma(W_f x_t + U_f h_{t-1} + P_f C_{t-1} + b_f) \tag{4.7}$$

$$\mathbf{C_t} = i_t\hat{C_t} + f_tC_{t-1} \tag{4.8}$$

$$\mathbf{o_t} = \sigma(W_ox_t + U_oh_{t-1} + P_oC_t + b_o) \tag{4.9}$$

$$\mathbf{h_t} = o_t \circ \ tanh(C_t) \tag{4.10}$$

$$\mathbf{y_t} = V_oh_t \tag{4.11}$$

4.5 Model 4: Bi-directional LSTM

The next model evaluated as part of this investigation is the Bi-directional LSTM (BLSTM). A limitation of the traditional RNN is that they can only incorporate the signal obtained from the pervious occurrences. However, in many natural language tasks, such as speech recognition, where an entire utterance is transcribed simultaneously, it can be beneficial to explore BLSTMs. In the case of AES, having knowledge of references made in the future (time$_{t+n}$) can be critical for deriving representations at time$_t$.

Before moving into the BLSTM, is it important to understand the structure of the simpler Bi-directional RNN (BRNN). As described by Schuster & Paliwal (1997), the BRNN contains two hidden layers, both of which are connect to the input and the output layers. One hidden layer processes information in the direction of the sequence, while the other processes information backwards along the sequence (See Figure 4.3).

The limitations of the BRNN is that it can only model information for where the end point is known, i.e not for online learning. However, in our particular case, AES, the model poses an interesting investigation. BRNN have achieved state-of-the-art results in tasks such as phoneme classifcation (Graves & Schmidhuber (2005)) and handwriting recognition (Graves et al. (2009)).

The hidden layers of the BRNN can be described by the following equations:

$$h_t = \sigma(U_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{4.12}$$

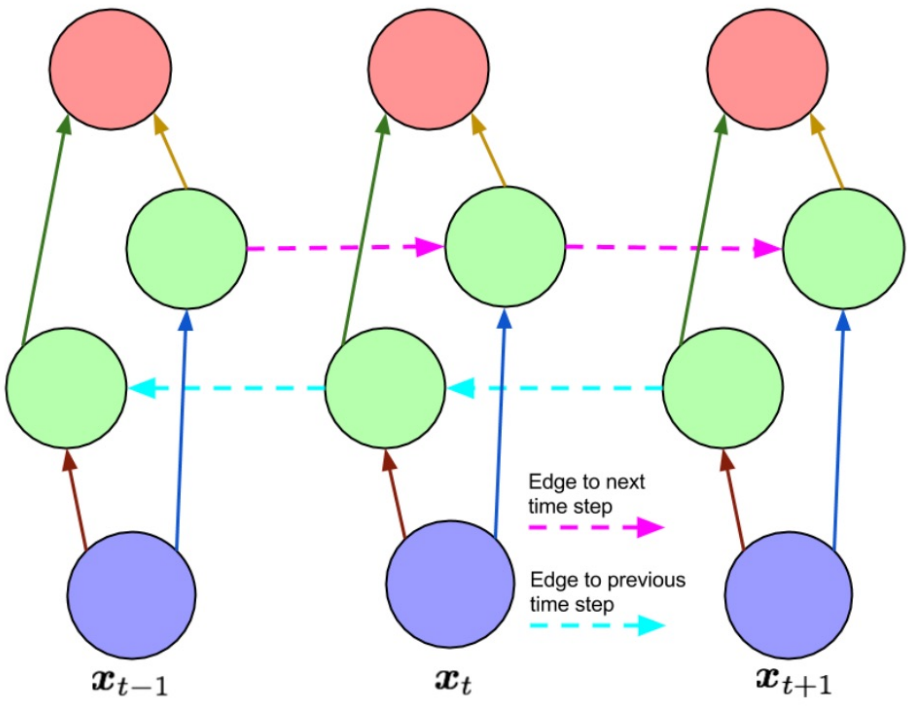

Figure 4.3: Figure showing Bi-directional RNN

$$z_t = \sigma(U_{zx}x_t + W_{zz}h_{t+1} + b_z) \tag{4.13}$$

$$\mathbf{y_t} = V_{oh}h_t + V_{oz}z_t + b_y \tag{4.14}$$

h_t is the hidden layer in the forward direction while z_t is the hidden layer in the backwards direction. The hidden layers are then added together at the output layer y_t to produce an combined representation. Combining BRNNs with LSTMs allows the model to capture signal for long sequences (characteristic of LSTMs in contrast to RNNs) and in both directions.

4.6 Model 5: Deep Bi-Directional LSTM

Much of the success in of neural networks can be attributed to **deep** architectures (Graves et al. (2013)). Deep models develop progressively higher levels of representation of the input data. In the case of a deep RNN, this is done by stacking multiple RNN hidden layers where the output of one hidden layer serves as the input of the next. The lower level layers are able to abstract the short term interactions between the inputs. The higher level layers represent interpretations spanning over longer areas of the input (Hermans & Schrauwen (2013)).

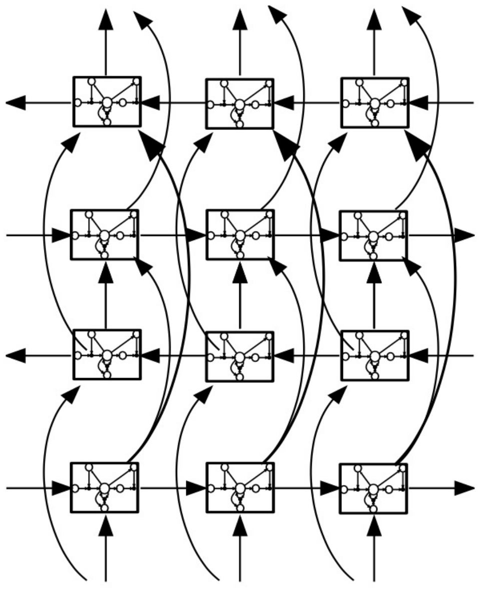

Figure 4.4: Figure showing deep bi-directional LSTM (Source: Graves & Schmidhuber (2005))

Applying the deep architectures to natural language sentences have shown to better capture the complexities and multi-scale effects characteristic of natural language (Irsoy & Cardie (2014)). Motivated by its performance on other natural language tasks (Graves et al. (2013)), we have decided to employ deep architectures to our AES system. In a deep RNN where each of the hidden layer functions are the same, the hidden layer can be formalised as follows:

$$h_t^n = tanh(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n) \tag{4.15}$$

$$y_t = W_{h^N y} h_t^N + b_y \tag{4.16}$$

Equation 4.16 is iteratively computed from $n = 1$ to N and $t = 1$ to T where the variables are the number of hidden layers and time respectively. In order to implement a Deep BRNN, the h^n can be replaced with a hidden layer that is computing the forward sequences and a hidden layer that is computing the backwards sequence (See Equation 4.12 and Equation 4.13). For a Deep BLSTM (as the one implemented for this investigation) one would simply replace the hidden layer with a BLSTM layer.

Chapter 5

Evaluation

This chapter aims to provide a detailed evaluation of the approach. It comprises of three experiments along with detailed discussions regarding each experiment. The experiments include: the hidden layer analysis; comparison of answer level versus scripts level; learning rate selection. The chapter also evaluates the performance of the five models as AES systems in general.

5.1 Experimental Setup

In order to build and evaluate the models, we used a numerical computation library in Python called Theano. It provides efficient functionality on CPUs and GPUs. Once the models were developed in Theano, Word2vec word embeddings were used to initialise the network. The values for the weight matrices were randomly initialised. Each model was trained for 50 epochs. At each epoch, an evaluation on the development set was carried out to observe the changes in Pearson, Spearman, and RMSE. The top performing models were evaluated against an unseen test set. The results section reports values for the unseen test set.

5.2 Results

5.2.1 Hidden Layer Analysis

The aim of this experiment is to determine the optimal hidden layer size for the models developed. We trained all five model on two hidden layer sizes $\{50, 100\}$. For models that require multiple hidden layers (i.e. RNN-MLP, BLSTM, DBLSTM), we assume that all hidden layers will have the same size. A learning rate lr 0.002 with the default RMSProp configurations were used to evaluate the models.

As can be seen in Table 5.1, RNN obtains of 0.242 and 0.305 for Pearson and Spearman respectively. It also achieves an RMSE of 5.922. Our hybrid model, RNN-MLP, shows improvements in the Pearson metric but experiences a slight drop in the Spearman score with values of 0.265 and 0.285 respectively. The RNN-MLP reports an RMSE value of 6.143.

Models	**Pearson**	**Spearman**	**RMSE**
RNN	0.242	0.305	5.922
RNN-MLP	0.265	0.285	6.143
LSTM	0.394	0.381	5.321
BLSTM	0.433	0.406	5.234
DBLSTM	**0.493**	**0.503**	**5.126**

Table 5.1: Table showing results for hidden-layer size **50**

Models	**Pearson**	**Spearman**	**RMSE**
RNN	0.291	0.282	5.750
RNN-MLP	0.301	0.347	6.033
LSTM	0.389	0.402	5.264
BLSTM	0.497	0.513	5.229
DBLSTM	**0.562**	**0.624**	**4.671**

Table 5.2: Table showing results for hidden-layer size **100**

LSTM with hidden layer size 50 reports values of 0.394, 0.381, and 5.321 for Pearson, Spearman, and RMSE respectively. Our BLSTM model reports values of 0.433 for Pearson, 0.406 for Spearman and 5.321 for RMSE. DBLSTM reports the top metrics with hidden layer size of 50 with Pearson, Spearman, and RMSE values of 0.493, 0.503, and 5.126 respectively.

Models	**Pearson**	**Spearman**	**RMSE**
RNN	+0.049	-0.023	-0.172
RNN-MLP	+0.036	+0.062	-0.110
LSTM	-0.005	+0.020	-0.057
BLSTM	+0.064	+0.107	-0.005
DBLSTM	+0.069	+0.121	-0.455

Table 5.3: Table showing difference observed between 50 and 100 hidden layer size

Table 5.2 reports values for all the models when configured with a hidden size of length 100. The change observed by increasing the size of the hidden layer can be seen in Table 5.3. RNN showed an improvement in Pearson, but suffered a penalty of -0.023 in the Spearman score. The RNN-MLP shows improvements in Pearson, Spearman, and RMSE score with values of 0.301, 0.347, and 6.033 respectively. The LSTM model shows a marginal performance decrease with a larger hidden layer (-0.005). BLSTM and DBLSTM model shows substantial improvements in all metrics with an increased hidden layer size. DBLSTM reports changes to Pearson, Spearman and RMSE with values of +0.069, +0.121, and -0.455 respectively.

5.2.2 Discussion: Hidden Layer Analysis

Deciding on the right hidden layer for your model can make a significant difference. Motivated by Hernández-Lobato & Adams (2015) we decided to start our experimentation with a hidden layer size of 50. However, is it often found that larger hidden layers perform better with more complex tasks Korattikara et al. (2015) and therefore we decided to also run an experiment with all the models on hidden layer size of 100. In the case where there is more than one hidden layer, each hidden layer is assumed to be the same size. Based on the results in Table 5.1 and Table 5.2, we can see that larger hidden layer size are indeed better for capturing more complex problems. However, it is worth noting that with larger hidden layer sizes have more parameters to compute and therefore are substantially more expensive. Furthermore, increase the size of the hidden layer too much can result in the model overfitting to the training data resulting in low performance when evaluated against unseen data.

5.2.3 Answer Level versus Script Level

Based on the results from Section 5.2.1 in determining the optimal hidden layer size, we decided to focus on optimising the top performing model (DBLSTM). As discussed in Section 4.1.2, script level essays are composed of answer level essays. The aim of this experiment is to determine whether smaller individual answer level essays can be more effective in regressing essay scores in comparison to larger script level essays.

The answer level scripts reported lower Pearson, Spearman and RMSE score with values of 0.582, 0.535, and 5.013 respectively. This was trained on a hidden layer size of 100 and learning rate lr of 0.002. Default RMSProp configuration can be assumed.

Level	**Pearson**	**Spearman**	**RMSE**
Answer Level	0.582	0.535	5.013
Script Level	0.624	0.562	4.671

Table 5.4: Table showing evaluation score between answer level texts and script level texts

5.2.4 Discussion: Answer Level versus Script Level

Constructing a hypothesis on the performance of script level texts in comparison to answer level texts was a difficult task. As answer level scripts contain less data to derive features discriminative of writing quality, our initial inclination would be in favour of script level texts. However, script level texts are simply an aggregation of at least one answer level script. For example if *answer 1* received a score of 20 and *answer 2* received a score of 4, then the script level score would report 24. The text contained in half the script level text will be disproportional to the other half.

The results in Table 5.4 show that in fact script level texts did produce better results than answer level texts. This can be attributed to the additional sequences provided by the script level. Additionally, the previous claim regrading the disproportional text, although possible, is unlikely given the same student complete both answers. Our results are further supported by the foundations of machine learning which supports the notion that models generally perform better if they receive more training data.

5.2.5 Learning Rate Selection

Selecting the right learning rate for your model can make a significant difference on the overall performance of your model. Choosing a rate that is too high can result in overshooting and never reaching the local minima. Alternatively, choosing a rate that is too low, can require a long time to converge (computationally expensive). As a result, we experiment with a variety of learning rates to identify the one that is optimal for our models. Having already identified the top performing model through our previous experiments, we have decided to report the results of our learning rates only on the DBLSTM with script level texts.

A learning rate of 0.01 results in the lowest performance with Pearson, Spearman and RMSE scores of 0.574, 0.523, and 5.238 respectively. Reducing the learning rate by a factor of 10 (0.001) improves Pearson by +0.017 and reduces the RMSE by -0.25. The top performing learning rate is 0.002 which reports 0.624, 0.562 and 4.671 for Pearson, Spearman and RMSE respectively. However, it is possible over a large number of epochs, the 0.001 learning rate may outperform.

lr	**Pearson**	**Spearman**	**RMSE**
0.01	0.574	0.523	5.238
0.002	0.624	0.562	4.671
0.001	0.591	0.545	4.988

Table 5.5: Table showing results for various learning rates *lr* {0.01, 0.002, 0.001}

5.2.6 Learning Rate Analysis

Our initial learning rate, 0.001, was set based on the recommendation of Tieleman & Hinton (2012). We also ran experiments with 0.01 and 0.002. It was found that 0.002 produced the best results for our particular model. The worst results were produced by the learning rate of 0.01. This can be a result of overshooting, despite the implementation of SGD optimiser RMSProp. For our particular epoch length (50), the learning rate of 0.002 performed slightly better than 0.001. Primarily because the steps are marginally larger and therefore was able to reach closer to the local minima in the given number of epochs.

5.2.7 Model Evaluation

Comparing the models in Table 5.1 and Table 5.2 at each iterative extension onto the RNN models showed improvements. The performance increase from RNN to RNN-MLP is indicative of the higher level complexities that can be abstracted from an additional layer. However, the absence of a feedback loop in the MLP hidden layer limits the potential signal that could be exploited from a higher level hidden layer. The architecture of the models is such that the entire essay is provided as a single input. When trying to capture discriminative writing features, such as flow of sentences, an aggregate view of all the sentences is needed. In order to do this, the model needs to maintain sequential information for longer time steps (characteristic of LSTMs).

LSTM's ability to capture longer sequence is evidenced by the higher evaluation results achieved. This is in line with the findings in literature, Irsoy & Cardie (2014); Graves et al. (2004), Eck & Schmidhuber (2002). However, as supported by Graves et al. (2013), the addition of bi-directionality, in the BLSTM model, provides an additional boost in performance. This can be attributed to the natural structure of language and the way we visually perceive information.

AES Models	**Pearson**	**Spearman**	**RMSE**
BLSTM (Zaidi, 2016)	0.624	0.562	4.671
SVM$_{rank}$ (Yannakoudakis et al., 2011)	0.741	0.773	n/a

Table 5.6: Table showing results obtained by our top model in comparison to the current state-of-the-art

When reading, we continually make eye movements called *saccades*. Saccades are not always forward moving. As a matter of fact, around 15% of saccades are regressions (right-to-left movements)(Rayner (1998)). Short regressed saccades are usually when information is not processed, while longer regressed saccades are due to the reader not understanding the text (Rayner (1998)). This phenomenon has been emulated in RNNs through the introduction of bi-directional networks. When processing text using neural networks, incorporating the bi-directionality aspect can facilitate the processing and understanding of text. As a matter of fact, this is evidenced by the results shown in Table 5.1 and Table 5.2 as well as other sources in literature (Graves & Schmidhuber (2005), Schuster & Paliwal (1997)).

Much of the recent success associated imaging and neural networks can be attributed to deep learning, or the inclusion of additional hidden layers. This is due to the complex patterns inherent in images and object detection; natural language is no different. Lower level hidden layers process text patterns in close proximity. The high level layers process data across a wider span of text and are capable identifying elaborate patterns. Combining the deep architecture with bi-directionality in our DBLSTM model, we have boosted the performance of our neural network substantially (See Table 5.2). The appreciation of Pearson and Spearman values and decrease in the RMSE metric shows that our model is indeed learning the complexity of scoring the AES better than the shallower less intricate models. However, as shown in Table 5.6, we are still a few iterations away from outperforming the current state-of-the-art.

Chapter 6

Visual Evaluation and Error Analysis

Transparency is an important aspect of implementing AES systems. Understanding how the AES system works is a vital ethical and commercial concern. Although users should have the ability to understand how the AES assign a particular grade, it should be in such a way that discourages reverse engineering and "teaching to test" . This chapter aims to build on our existing evaluation in the previous chapter by showing the inner workings of the models. The chapter will cover a brief overview of our method of visualisation. Additionally, it will provides examples and analysis of visual outputs made by our LSTM and DBLSTM models.

6.1 Visualisation Approach

In order to tackle these critical issues, we have attempted to provide more insight into the system through the use of visualisations. This evaluation technique aims to visually capture the networks performance by measuring the quality of individual word vectors. This work has been inspired by the work of Alikaniotis et al. (2016). When the model reaches its optimal performance, a single essay is passed through the system, one word[1] at a time. When the word reaches the end of the system, the total gradient of the error is calculated using backpropagation. However, the weight parameters are not updated. Instead, *pseudo targets scores* are provided to the error or cost function. The cost function can be defined as follows:

$$cost = \sum (y_i - \hat{y}_i)^2 \tag{6.1}$$

In the place of the target value, y_i, two *pseudo* targets scores are provided. The *max-pseudo* target score and the *min-pseudo* target score which are 40 and 0 respectively.

$$loss_{max} = \sum (40 - \hat{y}_i)^2 \tag{6.2}$$

$$loss_{min} = \sum (0 - \hat{y}_i)^2 \tag{6.3}$$

Calculating the gradient for $loss_{max}$ and $loss_{min}$ with respect to all of the parameters in the model will give us the relative quality our word embedding and whether it is positively and negatively influencing the score of our essay. In order to present a combined score ($score_c$), we perform the following:

$$score_c = loss_{min} - loss_{max} \tag{6.4}$$

The intuition behind the metric $score_c$ is that the higher the score the better the word. The quality of word have been divided into four levels {lowest, low, high, highest}. The quality of words are essay dependant. Therefore, $score_c$ between two different essays are not comparable. In order to categorise the quality of vectors the follow mathematical manipulations were carried out:

$$bin = [max(score_c) - min(score_c)]/4 \tag{6.5}$$

[1] *Please note that "word" for the purposes of this section, refers to any token in the essay including punctuations*

$$lowest = if min <= score_c < min + bin \tag{6.6}$$

$$low = if min + bin <= score_c < min + (bin * 2) \tag{6.7}$$

$$high = if min + (bin * 2) <= score_c < min + (bin * 3) \tag{6.8}$$

$$highest = if min + (bin * 3) <= score_c < min + (bin * 4) \tag{6.9}$$

6.2 Visualisation Examples

Below are some visualisations of sentences that have been processed by the LSTM and DBLSTM models. The quality of the vectors have been represented with the following colours: lowest quality vectors, low quality vectors, high quality vectors, and highest quality vectors.

Example 1

LSTM: Mr. Manager my name James Camirez and i'am writing this letter to you because I have some compleints for the disappointing evening I had last night.

DBLSTM: Mr. Manager my name James Camirez and i'am writing this letter to you because I have some compleints for the disappointing evening I had last night.

Looking at example 1 we can see that the model was able to detect that there is a missing verb ("is") before "James" and therefore James gets a lower score. Additionally it identifies the two misspelled words in the sentence, "I'am" and "compleints". The model was able to capture the correct use of the phrase "I have". In the context of non-native English speaks, the phrase "I have" can be quite ambiguous as it is usually associated with tangible objects. The correct use of a period at the end of the sentence is also awarded with a high quality vector.

Example 2

RNN: …International Arts Festival. This festival was well-orginised. There were a lot of activities for everyone. I relly liked exhibitions. I had a chance to see amazing pictures with wezy painted by Russian painters.

DBLSTM: …International Arts Festival. This festival was well-orginised. There were a lot of activities for everyone. I relly liked exhibitions. I had a chance to see amazing pictures with wezy painted by Russian painters.

Example 2 DBLSTM illustrates the models ability to capture correctly used references. Specifically, the phrase "This festival" is referring to "International Arts Festival". Using variable references for the same event is a sign of higher writing quality. As a result, the system awards this occurrence. The model picks up on what seems to be a grammatically incorrect sentence "…to see amazing pictures with wezy painted by Russian painters." The word "pictures" is given a low score. It seems that the system does not recognise that "wezy" could be a noun and therefore interprets the sentence as "pictures with <unknown> painted by Russian painters". Therefore the ambiguity can be detected around the word "pictures". A better model would be able to detect that "wezy" was a noun and ameliorate the score for "pictures". The term "by" in the phrase "painted by Russian painters" is correctly award a high score.

In the case of the LSTM, the model was not able to catch the spelling mistake "well-orginised". Additionally, it places less of a penalty of spelling "really" as "relly". The LSTM model penalised the essay score for using "This" at the beginning of the second sentence despite being discriminative of good writing quality. This illustrates the additional level of complexity that can be obtained by developing a "deep" architecture.

Example 3

LSTM/DBLSTM: Dear Mr Robertson , I am writing to express my pleasure about the successful programme that you have prepared…

Both models scored the word vectors similarly. "Mr" was penalised due to the missing period. The comma after "Dear Mr Robertson" is indicative of good writing quality. Interestingly the word "pleasure" was identified as a low quality vector. "Pleasure" is a B1-Level word and should not be considered a low quality vector. Both DBLSTM and LSTM fail to rectify this limitation.

Example 4

LSTM:…excellent idea to visit the National Art Gallery.

DBLSTM:…excellent idea to visit the National Art Gallery.

In example 4, we can see that the LSTM positively rates the word "visit". Its seems that the LSTM is exaggerating the positivity of vectors that should have less of an impact on the overall score. This is resolved with the DBLSTM which lowers the weighting of "visit" from *highest* to *high*. The terms "National" and "Gallery" have been highlighted because of their change in classification from the LSTM to the DBLSTM. This is due to the fact that the categorisation thresholds are very steep. If a vector is even 0.001 below the range for that particular category, it will be dropped to the lower ranking.

Example 5

LSTM: In addition, I would like to inform you about an advertisement…

DBLSTM: In addition, I would like to inform you about an advertisement…

Similar to example 4, in example 5 the AES system has rewarded the correct use of the transitional phrase "In addition". However, the LSTM has placed undue positive weighting which is corrected by the DBLSTM. Both models fail the recognise and reward the correct use of "an". Something that could perhaps be captured with additional training data.

Example 6

LSTM: We are really interested in latest fashions and new…

DBLSTM: We are really interested in latest fashions and new…

In example 6, both LSTM and DBLSTM fail to identify a short-distance subject-verb agreement case. The model has shown that it can reward correctly subject-verb agreement but failed to do so in this case. The DBLSTM correctly identified the missing determiner ('the') before"latest". This was not captured by the LSTM. The implementation of bi-directionality may have facilitated with this particular example. For example the phrase "...to death *in latest* shocking murder case" is an except from a news story in the Telegraph. A bi-directional model would positively rate this phrase. However, once the phrase "shocking murder case" is replaced with "fashions and...", then the neural network should penalise the score as it is not grammatically correct.

Chapter 7

Summary and Conclusions

The purpose of this investigation was to develop an AES system that demonstrates **meaningfulness**, **transparency**, and **robustness**. Through the use of neural networks and deep learning we have laid the foundation for future AES systems. RNNs and it's variants we have shown a promising start towards developing a state-of-the-art model. The architecture implemented as part of this investigation extracts features that cannot be reverse engineered. They also provide meaningful results when evaluated against the domain standard metrics (Pearson, Spearman, RMSE). Through the use of our visualisation tool, users can get insight into the inner workings of the "blackbox" model. This was demonstrated by calculating error gradients for each words with *pseudo* target scores. It was found that many complex features discriminative of writing quality were captured through our implementation.

The top performing model was the DBLSTM. It's performance can be attributed to the additional hidden layers that capture more complex patterns. Furthermore, the presence of bidirectionally in the system, allowed the model to pick up on additional errors that would otherwise not been extracted. Revisiting the initial aims set out for this investigation, we have managed to demonstrate that our system is **meaningful**, **transparent**, and **robust**. However, there need to be additional work in the area of improve human-machine inter-rater agreement. Although, the top performing model successfully captured spelling mistakes, subject-verb agreement errors, misspelled words, and incoherent phrases, it was not able to do so on a consistent basis.

Future work in this area includes integrating Bayesian methods of determining hyperparamters. This may provide an additional boost to the performance of our AES system. Relying on intuition and grid search approach is not a feasible solution to reach optimal results. Additionally, adopting a deep compositional approach where the layered model is training sentence level representations and essay level representations on different levels of the network can be an interesting area of investigation. Finally, although literature suggests that initialising the network using Word2vec improves convergence speed, Word2vec is trained for context. An interesting avenue to explore would be to train new embeddings that capture writing quality.

Bibliography

Attali, Y. & Burstein, J. (2006), 'Automated essay scoring with e-rater® v. 2', *The Journal of Technology, Learning and Assessment* **4**(3).

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F. & Gauvain, J.-L. (2006), Neural probabilistic language models, *in* 'Innovations in Machine Learning', Springer, pp. 137–186.

Briscoe, T., Medlock, B. & Andersen, Ø. (2010), 'Automated assessment of esol free text examinations', *University of Cambridge Computer Laboratory Technical Reports* **790**.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P. & Robinson, T. (2013), 'One billion word benchmark for measuring progress in statistical language modeling', *arXiv preprint arXiv:1312.3005* .

Chen, Y.-Y., Liu, C.-L., Lee, C.-H., Chang, T.-H. et al. (2010), 'An unsupervised automated essay scoring system', *IEEE Intelligent systems* **25**(5), 61–67.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), 'Learning phrase representations using rnn encoder-decoder for statistical machine translation', *arXiv preprint arXiv:1406.1078* .

Collobert, R. & 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', ACM, pp. 160–167.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011), 'Natural language processing (almost) from scratch', *The Journal of Machine Learning Research* **12**, 2493–2537.

Duchi, J., Hazan, E. & Singer, Y. (2011), 'Adaptive subgradient methods for online learning and stochastic optimization', *The Journal of Machine Learning Research* **12**, 2121–2159.

Eck, D. & Schmidhuber, J. (2002), Finding temporal structure in music: Blues improvisation with lstm recurrent networks, *in* 'Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on', IEEE, pp. 747–756.

Gers, F. (2001), Long short-term memory in recurrent neural networks, PhD thesis, Universität Hannover.

Gers, F. A. & Schmidhuber, J. (2001), 'Lstm recurrent networks learn simple context-free and context-sensitive languages', *Neural Networks, IEEE Transactions on* **12**(6), 1333–1340.

Graves, A., Eck, D., Beringer, N. & Schmidhuber, J. (2004), Biologically plausible speech recognition with lstm neural nets, *in* 'Biologically Inspired Approaches to Advanced Information Technology', Springer, pp. 127–136.

Graves, A., Jaitly, N. & Mohamed, A.-r. (2013), Hybrid speech recognition with deep bidirectional lstm, *in* 'Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on', IEEE, pp. 273–278.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. & Schmidhuber, J. (2009), 'A novel connectionist system for unconstrained handwriting recognition', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **31**(5), 855–868.

Graves, A. & Schmidhuber, J. (2005), 'Framewise phoneme classification with bidirectional lstm and other neural network architectures', *Neural Networks* **18**(5), 602–610.

Hermans, M. & Schrauwen, B. (2013), Training and analysing deep recurrent neural networks, *in* 'Advances in Neural Information Processing Systems', pp. 190–198.

Hernández-Lobato, J. M. & Adams, R. P. (2015), 'Probabilistic backpropagation for scalable learning of bayesian neural networks', *arXiv preprint arXiv:1502.05336* .

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Irsoy, O. & Cardie, C. (2014), Opinion mining with deep recurrent neural networks., *in* 'EMNLP', pp. 720–728.

Klebanov, B. B. & Flor, M. (2013), Word association profiles and their use for automated scoring of essays., *in* 'ACL (1)', pp. 1148–1158.

Korattikara, A., Rathod, V., Murphy, K. & Welling, M. (2015), 'Bayesian dark knowledge', *arXiv preprint arXiv:1506.04416* .

Larkey, L. S. (1998), Automatic essay grading using text categorization techniques, *in* 'Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 90–95.

McNamara, D. S., Crossley, S. A., Roscoe, R. D., Allen, L. K. & Dai, J. (2015), 'A hierarchical classification approach to automated essay scoring', *Assessing Writing* **23**, 35–59.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J. H. & Khudanpur, S. (2011), Extensions of recurrent neural network language model, *in* 'Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on', IEEE, pp. 5528–5531.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L. & Cernocky, J. (2011), Rnnlm-recurrent neural network language modeling toolkit, *in* 'Proc. of the 2011 ASRU Workshop', pp. 196–201.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, *in* 'Advances in neural information processing systems', pp. 3111–3119.

Mnih, A. & Hinton, G. (2007), Three new graphical models for statistical language modelling, *in* 'Proceedings of the 24th international conference on Machine learning', ACM, pp. 641–648.

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H. & Bryant, C. (2014), The conll-2014 shared task on grammatical error correction., *in* 'CoNLL Shared Task', pp. 1–14.

Page, E. B. (1967), Grading essays by computer: Progress report., *in* 'Proceedings of the invitational Conference on Testing Problems'.

Page, E. B. (1968), 'The use of the computer in analyzing student essays', *International review of education* **14**(2), 210–225.

Rayner, K. (1998), 'Eye movements in reading and information processing: 20 years of research.', *Psychological bulletin* **124**(3), 372.

Rudner, L. M. & Liang, T. (2002), 'Automated essay scoring using bayes' theorem', *The Journal of Technology, Learning and Assessment* **1**(2).

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1988), 'Learning representations by back-propagating errors', *Cognitive modeling* **5**(3), 1.

Schuster, M. & Paliwal, K. K. (1997), 'Bidirectional recurrent neural networks', *Signal Processing, IEEE Transactions on* **45**(11), 2673–2681.

Socher, R., Bauer, J., Manning, C. D. & Ng, A. Y. (2013), Parsing with compositional vector grammars., *in* 'ACL (1)', pp. 455–465.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y. & Potts, C. (2013), Recursive deep models for semantic compositionality over a sentiment treebank, *in* 'Proceedings of the conference on empirical methods in natural language processing (EMNLP)', Vol. 1631, Citeseer, p. 1642.

Somasundaran, S., Burstein, J. & Chodorow, M. (2014), Lexical chaining for measuring discourse coherence quality in test-taker essays., *in* 'COLING', pp. 950–961.

Tai, K. S., Socher, R. & Manning, C. D. (2015), 'Improved semantic representations from tree-structured long short-term memory networks', *arXiv preprint arXiv:1503.00075* .

Tang, D. (2015), Sentiment-specific representation learning for document-level sentiment analysis, *in* 'Proceedings of the Eighth ACM International Conference on Web Search and Data Mining', ACM, pp. 447–452.

Tieleman, T. & Hinton, G. (2012), 'Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude', *COURSERA: Neural Networks for Machine Learning* **4**, 2.

Turian, J., Ratinov, L. & Bengio, Y. (2010), Word representations: a simple and general method for semi-supervised learning, *in* 'Proceedings of the 48th annual meeting of the association for computational linguistics', Association for Computational Linguistics, pp. 384–394.

Yannakoudakis, H., Briscoe, T. & Medlock, B. (2011), A new dataset and method for automatically grading esol texts, *in* 'Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1', Association for Computational Linguistics, pp. 180–189.