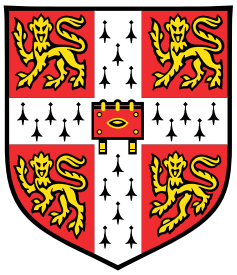
Interpretability for Deep Learning



Sanchit Gandhi

Supervisor: Prof. Mark Gales

Department of Engineering
University of Cambridge

This report is submitted for the degree of
Master of Engineering

Queens' College September 2021

This thesis reports on research supported by Cambridge Assessment, University of Cambridge.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this report are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This report is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This report contains fewer than 12,000 words and 60 pages including appendices, bibliography, footnotes, tables and equations.

Sanchit Gandhi
September 2021

Acknowledgements

I would like to thank Prof. Mark Gales for his guidance and support throughout this project. Weekly supervisions were always interesting, challenging and enlightening. As a result, I have learnt so much about this subject matter, and also research as a whole. For this, I am incredibly grateful to Mark. I would also like to thank Yiting (Edie) Lu for her all efforts in helping me get set up, running me through the preliminary models, checking theory and debugging errors along the way.

This year has been a particular challenge from a health point of view. A fourth-year Master's course is difficult in itself, but is made considerably harder by long-term illness. I am indebted to the unconditional support of my family and friends. Without them, completion of this work would not have been possible.

Abstract

Deep neural networks (DNNs) are well-known for their high performance in various machine learning and artificial intelligence tasks. However, it is often difficult for these networks to yield explanations for their predictions in terms that humans can easily understand. Due to their highly-distributed and non-linear nature, the features from which they draw conclusions can be so complex and abstract that it is a challenge to establish why a model produces the predictions that it does. This project aims to address this issue, by improving understanding of the representations DNNs learn for a given task. Specifically, the focus is on the application of DNNs to the machine translation (MT) task of spoken language 'grammatical error correction' (GEC). The overriding aim of the project is to find neurons, or combinations of neurons, in a particular layer of a model that are responsible for the error correction of certain parts of speech or types of grammatical error. These neurons are utilised to control the system translation: their activations are varied and the effect on the network's predictions assessed.

Uncovering insights into the operation of DNNs provides a basis for developing methods relating to network control, a means by which network predictions can be modified in a predictable and reliable way. By default, deep learning models pick up biases from training data. Network control provides a mechanism for correcting such biases. Furthermore, controllable models may be made to yield output translations with properties that closely match low-resource existing data, serving the purpose of data augmentation, or fine-tuned to match out-of-domain (OOD) datasets, allowing for domain adaptation.

The first interpretabiltiy analysis explores whether individual neurons influence particular aspects of the system. It investigates the similarity between representations learned by neural networks with the same architecture trained from different random initialisations. This is achieved by correlating learned neuron representations in one network to those learned by neurons in a separate independently trained network. Two algorithms are proposed to rank neurons based on their correlation scores. It is shown experimentally that translation quality is highly dependent on the discovered neurons. An attempt is made at finding individual nodes responsible for part of speech (PoS) correction. Modifying the activations

of individual neurons is demonstrated to be an ineffective way of controlling GEC translation- the translation is impacted globally across many parts of speech.

The analysis is generalised to find combinations of neurons, or network directions, which act together to yield network representations. A gradient-based approach is adopted, which finds the combination of neurons that most greatly impacts the system's predictions. The experimental results show the gradient direction to be an effective way of modifying the system's output. These findings prompt the subsequent investigations, which seek to uncover directions that strongly influence GEC for particular parts of speech.

The network node correlation approach is refined to correlating learned direction representations in one network to learned direction representations in a separate independently trained network. This is achieved through the use of Singular Vector Canonical Correlation Analysis (SVCCA), a modified version of Canonical Correlation Analysis (CCA), which first uses a Singular Value Decomposition (SVD) to remove low variance noise directions. The directions found by SVCCA are disproportionately important to the representation learned by a layer, relative to the neuron-aligned directions. Modifying the extent to which these directions are expressed is shown to be an effective way of controlling the GEC system for particular parts of speech.

A different approach to finding network directions involves building a logistic classifier to predict the PoS associated with a particular node or combination of nodes. This method is based on the assumption that if a classifier of simple architecture can predict the PoS label assigned to a node, then that node implicity encodes the learned information for that PoS. By construction, there is a clear interpretation of the learned classifier parameters, and by extension the representations of parts of speech in the GEC system. Two ways are presented in which the classifier's parameters can be used for model control. For both, the system's predictions are altered for a given PoS and not others.

A task-based approach is adopted to compare and contrast the network directions returned by the various interpretability analyses. The GEC systems are trained on written text data. An attempt it made at fine-tuning the GEC system to OOD transcribed speech data, which has different statistics of PoS errors. The task is then to learn the scale of the directions required to reflect the difference in PoS statistics, thus adapting the model to the speech data. Network directions of greater saliency result in higher translation performance on the speech data following fine-tuning. Compared to the baseline system, this fine-tuning process yields improvements in predictive performance. However, the gains are somewhat low. Further investigations are made using artificial data sets to better understand why this might be the case. A number of theoretical arguments are presented explaining potential reasons why the fine-tuned system performance is not greater.

Table of contents

Chapter 1

Introduction

1.1 Motivation

In the last decade, the application of *deep neural networks* (DNNs) to long-standing problems in the fields of speech recognition and machine translation has brought about breakthroughs in performance and prediction power [13, 23]. Though these systems yield excellent performance, it is often challenging to understand how and why network decisions are made. Many of these models behave as black-boxes, failing to provide explanations for their predictions. This issue arises due to the essence of DNNs; the highly distributed nature of information and non-linearity of networks means they operate in a space which does not correspond directly to high-level concepts readily understood by humans [18]. However, it is these same two features which enable DNNs to perform so well.

This project aims to address the problem of understanding by developing approaches for neural network *interpretability*, defined as the degree to which a human can understand the cause of a system's decision [24]. The particular application is neural machine translation (NMT) approaches for spoken language 'grammatical error correction' (GEC). This task involves using a DNN to generate an output sequence given an input sequence, where the two sequences may be of different lengths. The purpose of a GEC system is to take an input sentence which contains grammatical errors and output a corrected sentence.

Spoken language GEC is an important mechanism to help learners of a foreign language improve their spoken grammar, particularly in the context of automatic assessment of second language acquisition in computer assisted language learning (CALL). Improving network interpretability for GEC paves the way for a number of speech and language processing applications.

There are large amounts of labelled written text data which can be used to train GEC models. However, due to the differences between written and spoken communication, these

text-based systems do not generalise well to speech transcripts. Written text tends to be more formal than spoken communication, with greater care taken over word choice, punctuation and structure. Furthermore, there are often disfluencies in spontaneous speech, caused by repetitions and hesitations [31], which are not found in written text. These differences alter the grammatical structure, which in turn affects the error correction process. Since there is little labelled speech data available, direct training is not possible.

Uncovering insights into the operation of these DNNs provides a basis for developing methods relating to network *control*. The intention of this project is to establish a framework by which the network's predictions can be modified in a predictable and reliable way. By constructing the control parameters which govern this process as a direct product of the findings of an interpretability analysis, they themselves have interpretable meanings and significances. In doing so, a GEC model can be trained on labelled written text data, of which there is an abundance, and then controlled to generate outputs with grammatical error corrections which closely match low-resource speech data, a procedure termed *data augmentation*. Large amounts of this artificial data can easily be produced, which can then go on to be used to directly train a GEC model, thus overcoming the issue of insufficient speech data.

Domain adaptation offers an alternative solution to the problem of a mismatch between source and target data. In this case, a GEC model is trained in the written text domain, followed by a form of transfer learning, in which the control parameters of the model are fine-tuned on the speech domain. Optimising just the control parameters returns substantial computation savings compared to directly adapting all the learnable parameters of the DNN [33]. Furthermore, due to their interpretable essence, the changes to parameter values are indicative of the modifications made to the model in fine-tuning from the source to target domain, very much unlike the parameters of the DNN.

1.2 Outline of Report

This report consists of 6 chapters. Chapter 2 introduces the types of deep learning approaches used for GEC and how interpretability analysis is performed for such systems. Chapter 3 describes how salient network nodes are identified and the impact of node perturbation on the GEC system. Chapter 4 extends this to more general network direction perturbations. Chapter 5 describes the form of the training and evaluation data sets, the experimental set-up, the experimental results and discusses their consequences. Finally, Chapter 6 draws the main conclusions of this project and presents possible future work.

Chapter 2

Deep Learning for GEC

The purpose of this chapter is to provide background knowledge about deep learning approaches for GEC. This chapter begins by briefly reviewing the style of sequence-sequence models that are applied to GEC. There are an array of other deep learning techniques feasible for this task. However, in this report, the focus is specifically on sequence-to-sequence models. The chapter then proceeds to describe how performance analysis is achieved for GEC systems. It concludes with an overview of domain adaptation and its meaning in the context of GEC.

2.1 Sequence-to-Sequence Models

Machine translation (MT) is the task of mapping one sequence into another sequence (sequence transformation). Neural machine translation (NMT) is a newly emerging approach to machine translation [17, 6]. Unlike traditional phrase-based statistical machine translation (SMT) systems [3, 20], which consists of many small sub-components that are tuned separately, NMT attempts to build and train a single, large DNN that reads an input sequence and outputs a correct translation.

Sequence-to-sequence models are a field of MT approaches that tackle the task of mapping an input sequence to an output sequence that is not necessarily of the same length. This comes up in many applications, such as speech recognition, machine translation or question answering, where the input and output sequences in the training set are generally not of the same length (although their lengths might be related).

From a probabilistic perspective, sequence-to-sequence translation is equivalent to finding a K-length target sequence $\boldsymbol{y}_{1:K}$ that maximises the conditional probability of $\boldsymbol{y}_{1:K}$ given a T-length source sequence $\boldsymbol{w}_{1:T}$, i.e., $\operatorname{argmax} P(\boldsymbol{y}_{1:K}|\boldsymbol{w}_{1:T})$, where the lengths K and T can vary from each other. In NMT, a parameterised model is trained to maximise the

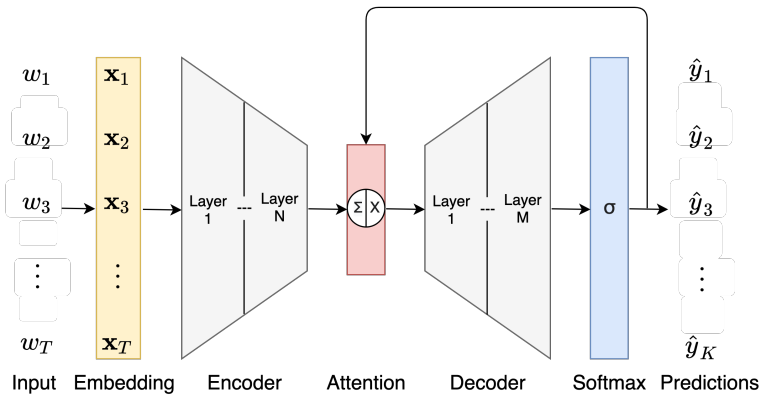


Figure 2.1 Sequence to sequence encoder-decoder RNN model.

conditional probability of source-target sequence pairs using a parallel training corpus. Once the discriminative model is trained, given a source sequence, a corresponding prediction can be generated by searching for the sequence that maximises the conditional probability.

There have been a number of proposed ways in which to use neural networks to directly learn this conditional distribution [5, 34]. The NMT approach used in this project follows the sequence-to-sequence learning framework presented in [36]. The model has three components: a recurrent neural network (RNN) encoder, an RNN decoder, and an attention mechanism. The encoder transforms a source sequence into a sequence of hidden vectors, one vector per input symbol. Given this sequence of vectors, the decoder predicts one symbol at a time, until the special end-of-sequence (EOS) symbol is produced. The encoder and decoder are connected through an attention module [1], which allows the decoder to focus on different regions of the source sentence during the course of decoding, helping the system cope effectively with long input sequences. The overall architecture, illustrated in Figure 2.1, is termed the encoder-decoder or sequence-to-sequence architecture.

The first stage in the encoder is a mapping from the t-th input word w_t to a real-valued vector $\boldsymbol{x}_t$ that encodes the meaning of the word. These word representations are often referred to as *word embeddings*. In this interpretation, the raw input words are viewed as points in space of dimension equal to the vocabulary size. As every word is represented by a one-hot vector, every pair of words is at Euclidean distance $\sqrt{2}$ from each other. The word representations embed those points in the original space in a feature space of lower dimension. In this embedding space, words that frequently appear in similar contexts (or any pair of words sharing some "features" learned by the model) are close to each other.

The hidden layer activation vector for the first layer is constructed using a bi-directional recurrent neural network (BiRNN) [30] with long short-term memory (LSTM) recurrent units [14]. A BiRNN consists of forward and backward RNN's. The forward RNN reads the

input sequence as it is ordered (from $\boldsymbol{x}_1$ to $\boldsymbol{x}_T$) and calculates a sequence of forward hidden states $(\overrightarrow{\boldsymbol{h}}^1_1, \ldots, \overrightarrow{\boldsymbol{h}}^1_T)$. The backward RNN reads the sequence in the reverse order (from $\boldsymbol{x}_T$ to $\boldsymbol{x}_1$), resulting in a sequence of backward hidden states $(\overleftarrow{\boldsymbol{h}}^1_T, \ldots, \overleftarrow{\boldsymbol{h}}^1_1)$. The hidden layer activation for the t-th input word is obtained by concatenating the forward hidden state $\overrightarrow{\boldsymbol{h}}^1_t$ and the backward one $\overleftarrow{\boldsymbol{h}}^1_t$. This enables the feature representation of each input word to summarize not only the preceding words, but also the following words.

$$\overrightarrow{\boldsymbol{h}}^1_t = \mathrm{LSTM}\left(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}^1_{t-1}\right) \tag{2.1}$$

$$\overleftarrow{\boldsymbol{h}}^1_t = \mathrm{LSTM}\left(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}^1_{t+1}\right) \tag{2.2}$$

$$\boldsymbol{h}^1_t = \mathrm{concat}\left(\overrightarrow{\boldsymbol{h}}^1_t, \overleftarrow{\boldsymbol{h}}^1_t\right) \tag{2.3}$$

The forward and backward hidden states are used as the inputs to the subsequent layer of the encoder, which itself is another BiRNN. *Residual connections* [12] are applied between layers of the encoder to improve the training efficiency of large DNNs:

$$\overrightarrow{\boldsymbol{h}}^n_t = \mathrm{LSTM}\left(\overrightarrow{\boldsymbol{h}}^{n-1}_t, \overrightarrow{\boldsymbol{h}}^n_{t-1}\right) \tag{2.4}$$

$$\overleftarrow{\boldsymbol{h}}^n_t = \mathrm{LSTM}\left(\overleftarrow{\boldsymbol{h}}^{n-1}_t, \overleftarrow{\boldsymbol{h}}^n_{t+1}\right) \tag{2.5}$$

$$\boldsymbol{h}^n_t = \mathrm{concat}\left(\overrightarrow{\boldsymbol{h}}^n_t, \overleftarrow{\boldsymbol{h}}^n_t\right) + \boldsymbol{h}^{n-1}_t \tag{2.6}$$

for $n = 2, \ldots, N$. The use of multiple layers enables the system to progressively extract higher level features from the raw input data [9]. Due to the tendency of RNNs to better represent more recent inputs, the hidden layer activations indexed at time t are more focused on the input words around w_t.

The decoder processes the sequence of encoded vectors to predict the sequence of output symbols. The decoder network comprises of M uni-directional LSTM layers, since it performs translation in just one direction, combined with context vectors that are obtained through an attention mechanism. During *training*, the target sentence $\boldsymbol{y}_{1:K}$ is available. It is mapped to the sequence of vectors $\boldsymbol{x}'_{1:K}$ using word embeddings, as in the encoder. This decoding method is termed *teacher forcing* [35].

$$\boldsymbol{d}^1_i = \mathrm{LSTM}\left([\boldsymbol{x}'_{i-1}, \boldsymbol{s}_{i-1}], \boldsymbol{d}^1_{i-1}\right) \tag{2.7}$$

$$\boldsymbol{d}^m_i = \mathrm{LSTM}\left(\boldsymbol{d}^{m-1}_i, \boldsymbol{d}^m_{i-1}\right) \quad \text{for } m = 2, \ldots, M \tag{2.8}$$

The context vector $\boldsymbol{c}_i$ is computed as a weighted sum of the hidden-states $\boldsymbol{h}_t^N$:

$$\boldsymbol{c}_i = \sum_{t=1}^{T} \alpha_{i,t} \boldsymbol{h}_t^N; \quad \alpha_{i,t} = \frac{\exp\left(e_{i,t}\right)}{\sum_{j=1}^{T} \exp\left(e_{i,j}\right)}; \quad e_{i,t} = \mathrm{att}\left(\boldsymbol{d}_i^M, \boldsymbol{h}_t^N\right), \tag{2.9}$$

where att is an attention mechanism or alignment model which scores how well the inputs around position t and the output at position i match. There are two standard forms for the attention function: *dot-product* attention $e_{i,t} = \left(\boldsymbol{h}_t^N\right)^T \boldsymbol{W}_{xq} \boldsymbol{d}_i^M$ or *additive* attention $e_{i,t} = \boldsymbol{w}^T \tanh(\boldsymbol{W}_x \boldsymbol{h}_t^N + \boldsymbol{W}_k \boldsymbol{d}_i^M)$. Let $\alpha_{i,t}$ be the softmax probability that the target word y_i is translated from the source word w_t. The approach of taking a weighted sum to find the context vector can then be interpreted as computing an expected hidden state, where the expectation is taken over all possible hidden states with probabilities $\alpha_{i,t}$.

The context vector and the decoder hidden state are passed through a feedforward layer:

$$\boldsymbol{s}_i = \tanh\left(\boldsymbol{W}_c \boldsymbol{c}_i + \boldsymbol{W}_d \boldsymbol{d}_i^M + \boldsymbol{b}_s\right). \tag{2.10}$$

To make the final prediction, $\boldsymbol{s}_i$ is passed into a feed forward network followed by a softmax layer, and a probability distribution over the entire vocabulary generated:

$$P(y_i \mid \boldsymbol{w}_{1:T}, \boldsymbol{y}_{<i}) = \mathrm{softmax}(\boldsymbol{W}_s \boldsymbol{s}_i + \boldsymbol{b}_P). \tag{2.11}$$

The prediction $\hat{y}_i$ is found by *sampling* from this distribution:

$$\hat{y}_i = \underset{y_i \in \mathcal{V}}{\mathrm{argmax}} \left\{P(y_i \mid \boldsymbol{w}_{1:T}, \boldsymbol{y}_{<i})\right\}, \tag{2.12}$$

where $\mathcal{V}$ is the NMT vocabulary. Outputting the most probable word at each step is called *greedy decoding*. At *inference* time, the targets $\boldsymbol{y}_{1:K}$ are not known. Hence, the word embedding for the previous prediction $\hat{y}_{i-1}$ is used instead of the previous target word y_{i-1} in the first layer of the decoder. This method of decoding is termed *free-running*.

During training, *dropout* [32] is employed to improved generalisation. For each training pass, a fixed proportion of nodes in the network are de-activated, preventing individual nodes specialising to a single task. In addition, the optimisation routine performed during training time is refined through use of *batch normalisation* [15]. Batch normalisation is a method of adaptive reparametrisation that attempts to make the distribution of nonlinearity inputs more stable as the network trains. This is achieved through normalisation steps that fix the means and variances of layer inputs over mini-batches of training data, thus keeping layer output values in the linear regions of their respective activation functions. Batch normalisation

cannot be performed dynamically at test time, as the system receives only one sequence sample at a time. Hence, following training, the expected normalisation terms are computed.

In this project, the interpretability and analysis is focused on a particular layer of the network: the encoder output layer. The output layer of the encoder performs the optimal encoder transformation of the input word sequence. It is optimal in the sense of supplying the maximum amount of information to the attention mechanism and decoder module. Furthermore, it is the last layer of the model for which each hidden-state can directly be linked back to the elements of the inputs; there is one hidden layer activation $\boldsymbol{h}_t$[1] for every input word symbol w_t. Beyond the encoder output layer, the model's operation becomes considerably more complex due to the attention mechanism, decoder and back-history. Hence, the encoder output layer is the natural choice of layer to explore.

2.2 Sequence-to-Sequence Models for GEC

Sequence-to-sequence models can be applied to the task of GEC through appropriate construction of the parallel training corpus. For GEC, the input sequences are (possibly) ungrammatical sentences. The target sequences are the corresponding grammatically correct sentences. It is assumed there is supervised training data $\mathcal{D}$, consisting of an input sequence of T words $\boldsymbol{w}_{1:T}$ and an K-length target sequence $\boldsymbol{y}_{1:K}$, from which the network parameters $\boldsymbol{\theta}$ are trained by assessing the network predictions $\hat{\boldsymbol{y}}_{1:L}$:

$$\mathcal{D} = \{\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:K}\}\,; \quad \hat{y}_i = \mathcal{F}\left(\boldsymbol{w}_{1:T}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}\right), \tag{2.13}$$

where $\mathcal{F}$ is a non-linear mapping performed by the sequence-to-sequence model. Thus, the system learns to map (possibly) grammatically erroneous sentences to grammatically correct sentences.

A clear interpretation of the input and target sequences as applied to GEC allows for task-specific realisations of the teacher forcing and free-running decoding schemes. The teacher forcing algorithm trains sequence-to-sequence GEC models by supplying the system with the ground truth output y_{i-1} as an input to the decoder at the i-th decoding step. By clamping the inputs to the decoder to the target sequence, the system is made robust to model history errors. Consider the example presented in Figure 2.2; the model predicts the word `cats` instead of `cat` as the second word. However, the ground truth of `cat` is input to the decoder at the next stage. Consequently, the model's subsequent predictions are unaffected

[1] This can be generalised to the consider the hidden layer activations for the n-th encoder layer $\boldsymbol{h}_t^n$. However, since this project treats only one particular layer of the network, the encoder output layer, this notation is dropped to make the expressions more terse.

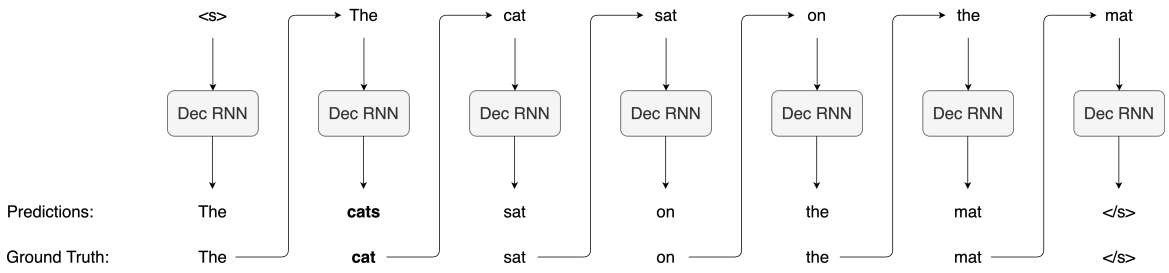


Figure 2.2 A sample sentence decoded using the teacher forcing decoding scheme. The ground truth outputs are supplied as inputs to the decoder.

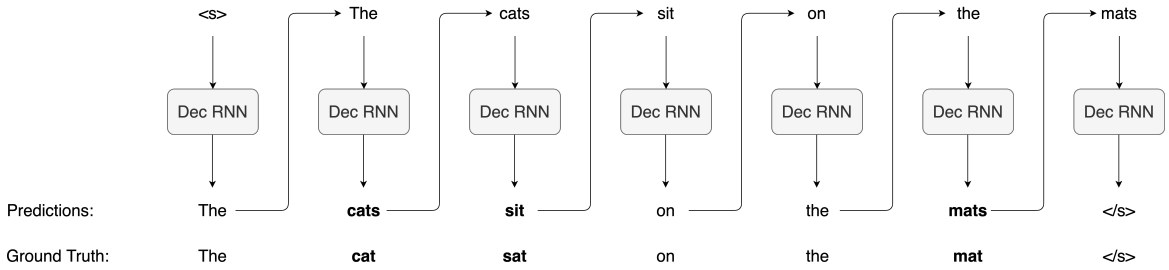


Figure 2.3 A sample sentence decoded using the free-running decoding scheme. The predicted outputs are supplied as inputs to the decoder.

by the back-history of predictions $\hat{\boldsymbol{y}}_{<i}$ and thus any previous incorrect predictions. For GEC, this means grammatical errors on the output sequence are attributed to the decoder operation for the i-th output only, and not its operation for previously decoded words. This enables for an effective interpretability analysis, as one can focus on instances where the model decodes grammatically incorrect terms and probe the system's operation at those points.

When the system is deployed at inference time, the true outputs are generally not know. In this case, the correct output y_{i-1} is approximated with the model's prediction $\hat{y}_{i-1}$, which is fed back into the model as the input to the decoder at the i-th time step. When the inputs are self-generated, the model is said to be in free-running mode. Using this decoding scheme, the prediction of later outputs is conditioned on previous outputs. Hence, an incorrect prediction results in the back-history of the predicted sequence being changed. This can result in a cascade effect of grammatical errors in the translation. Take the example shown in Figure 2.3. In teacher forcing mode, the model was made robust to the prediction of the word `cats` instead of `cat`. Whereas in free-running mode, this error is propagated to the subsequent predictions, resulting in two further incorrect outputs. This makes it considerably more difficult to localise where the GEC model's predictions go off-track. Thus, for an interpretability analysis, it is more difficult to ascertain what aspects of the model's operation caused it to deviate from the target sequence.

2.3 GEC Analysis

This section briefly describes the ways in which GEC systems can be analysed at an overall translation quality level and for different aspects of system performance.

2.3.1 Predictive Performance

The overall predictive performance of GEC systems is quantified through the use of the Generalized Language Evaluation Understanding metric (GLEU) [26], a variation of the Bilingual Evaluation Understudy (BLEU) metric commonly used in MT [28]. GLEU is n-gram precision metric in the range 0 to 1 (or 0 to 100 in terms of percentages) that automatically assesses the similarity between hypothesis and target sentences for GEC. Values closer to 1 (or 100 in terms of percentages) represent more similar texts.

Specifically, GLEU calculates a modified precision p_n by measuring the overlap between the hypothesis n-grams $H = \{\hat{y}_1, \ldots, \hat{y}_L\}$ and reference n-grams $R = \{y_1, \ldots, y_K\}$, rewarding hypothesis n-grams that overlap with the reference but not the original n-grams $O = \{w_1, \ldots, w_T\}$ ($R \backslash O$), and also penalising hypothesis n-grams that overlap with the original but not the reference ($O \backslash R$).

$$p_n = \frac{\left(\sum\limits_{ngram \in \{H \cap R\}} \text{count}_{H,R}(ngram) - \sum\limits_{ngram \in \{H \cap O\}} \max\left[0, \text{count}_{H,O}(ngram) - \text{count}_{H,R}(ngram)\right] \right)}{\sum\limits_{ngram \in \{H\}} \text{count}_H(ngram)} \tag{2.14}$$

where:

$$\text{count}_A(ngram) = \#\text{ occurrences of } ngram \text{ in } A$$
$$\text{count}_{A,B}(ngram) = \min(\#\text{ occurrences of } ngram \text{ in } A, \#\text{ occurrences of } ngram \text{ in } B)$$

A brevity penalty (BP) which penalises hypotheses of length L that are shorter than references of length K is calculated as:

$$\text{BP} = \begin{cases} 1 & \text{if } L > K \\ \exp(1 - K/L) & \text{if } L \leq K \end{cases} \tag{2.15}$$

The final GLEU score for parallel original, hypothesis and reference sequences is hence computed as:

$$\text{GLEU}(O, H, R) = \text{BP} \cdot \exp\left(\frac{1}{N} \sum_{n=1}^{N} \log p_n \right), \tag{2.16}$$

where the maximum size of the n-gram window N is typically set to 4.

GEC systems are often only evaluated in terms of overall performance because system hypotheses are not annotated by PoS or error types. Consequently, error correction information is lost. This is a major shortcoming, as analysis can only be performed at a very coarse level of granularity. A system that performs poorly overall may in fact outperform other systems at specific parts of speech or error types. However, the latter is unrevealed through use of GLEU score alone. The following two methods facilitate system performance at a finer level of granularity, through automatic annotation of one or both sides of the system's input and outputs.

2.3.2 Part of Speech Tagging

Part of speech tagging (PoS tagging) is the process of classifying words into their parts of speech and labelling them accordingly. Tags are assigned to words based on both their definition and their context. PoS tags are quintessential to the interpretability and analysis of GEC models. Accumulating PoS statistics reveals the distribution of parts of speech in a sequence of words. These statistics can then be used to query how the model is operating on a PoS by PoS basis.

In this project, PoS labels were generated for sequences of words using the **Stanford Neural Network Dependency Parser**[2], a piece of software that reads the input text and assigns a label from a specified PoS set to each word.

There are two distinct ways in which PoS tagging can be operated: on the input side of the system or on the output side. As explained in Chapter 2.1, the interpretability and analysis in this project is concentrated on the encoder output layer, the last layer of the model in which the hidden-states can be directly linked back to the input sequence. In addition, the hidden layer activation vector $\boldsymbol{h}_t$ is strongly affiliated to the input word w_t. Hence, labelling the input sequence with PoS tags effectively generates proxy PoS labels for the hidden layer activations. This is powerful, as it allows such questions to be asked as: which nodes are responsible for particular parts of speech on the inputs?

Table 2.1 A sample sentence, its corresponding PoS tags and hidden layer activation vectors. The PoS tags serve as proxy labels for the hidden layer activations.

Sentence:	the	cat	sit	on	the	mat
PoS Tags:	det.	noun	verb	prep.	det.	noun
Activation:	$\boldsymbol{h}_1$	$\boldsymbol{h}_2$	$\boldsymbol{h}_3$	$\boldsymbol{h}_4$	$\boldsymbol{h}_5$	$\boldsymbol{h}_6$

[2]Stanford Neural Network Dependency Parser: https://nlp.stanford.edu/software/nndep.html

In addition to the input sequence $\boldsymbol{w}_{1:T}$, PoS tags can be assigned on the output side to the target sequence $\boldsymbol{y}_{1:K}$. For a given data set, the input and target sequences remain fixed, meaning that the labels for the input and target sequences remain fixed, irrespective of the system's operation. By grouping the system's predictions by target PoS, one can assess the influence of the input side on the output predictions for a particular PoS. This enables such questions to be asked as: which nodes are responsible for encoding a PoS on the input to a PoS on the output?'

PoS tags can also be generated for the predicted sequence $\hat{\boldsymbol{y}}_{1:L}$. Unlike the input and target sequences, the predicted sequence may change as the model is altered. Thus, the input and target labels act as references in scenarios where the hypothesis sequence may be changing. The statistics of the PoS tags for the predictions can be compared to the PoS statistics for the fixed input and target sequences, revealing whether or not the model changes its corrections for particular parts of speech.

2.3.3 Annotation and Evaluation of Error Types

Rather than analysing one side of the system in isolation, the two can be linked. Error type evaluation is possible through annotation of the parallel original and corrected sentences with grammatical error type information. The first step in annotation is the alignment of the source and corrected sequences. This is necessary in order to map grammatical errors to corrections (sometimes referred to as 'correction detection'). The next step is identification of the grammatical error correction. The third and final step is classification of the type of error correction. In this project, the **grammatical ERRor ANnotation Toolkit** (ERRANT)[3], a toolkit that automatically extracts and classifies edits between original and modified sentence pairs, was utilised to perform these operations.

Table 2.2 A sample grammatically incorrect sentence and its corrected counterpart, annotated with grammatical error type information extracted and classified by ERRANT. The aligned words are shown in bold on the input and output sequences.

Input:	the	cat	**sit**	on	the	mat
Output:	the	cat	**sat**	on	the	mat
ERRANT:	-	-	verb	-	-	-

[3]ERRANT Toolkit: https://github.com/chrisjbryant/errant

2.4 Domain Adaptation

Domain adaptation refers to the situation where what has been learned in one setting is exploited to improve generalisation in another setting. The task (and the optimal input-to-output mapping) remains the same between each setting.

In the context of GEC, a DNN is trained in one domain (referred to as the source domain) which has a large amount of training data available, followed by a form of transfer learning in which the model parameters (or a sub-set of parameters) are fine-tuned on the domain of interest (referred to as the target domain). There might be discrepancies between the two domains in terms of the distribution of parts of speech or types of grammatical errors. Fine-tuning attempts to improve system performance on the target domain by adjusting the model parameters to reflect these differences. The task of automatically detecting and correcting grammatical errors in text remains the same between the two domains.

In this project, GEC systems are trained on written learner text data. The target domain of interest is transcribed learner speech data. Hence, the target domain can be interpreted as the speech domain.

Chapter 3

Network Node Perturbation

As discussed in the previous chapter, the interpretability and analysis in this project is focused on the final layer of the encoder. In order to facilitate this, the network's predictions are partitioned into two distinct stages. The first is a mapping from the sequence of input words $\boldsymbol{w}_{1:T}$ to the t-th hidden layer activation for the encoder output layer $\boldsymbol{h}_t$. The second mapping takes the sequence of hidden layer activations $\boldsymbol{h}_{1:T}$ and the $i-1$ previous predicted words $\hat{\boldsymbol{y}}_{<i}$ to form the network's prediction of the i-th output $\hat{y}_i$:

$$\boldsymbol{h}_t = \mathcal{F}_{h,t}\left(\boldsymbol{w}_{1:T}; \boldsymbol{\theta}\right); \quad \hat{y}_i = \mathcal{F}_y\left(\boldsymbol{h}_{1:T}, \hat{\boldsymbol{y}}_{<i}; \boldsymbol{\theta}\right). \tag{3.1}$$

If the system is run in teacher-forcing, then the $i-1$ previous target words $\boldsymbol{y}_{<i}$ are used in-place of the predicted words $\hat{\boldsymbol{y}}_{<i}$. The specific form of analysis involves perturbing the hidden layer activation of the encoder output layer by an amount $\Delta\boldsymbol{h}_t$ and evaluating the impact on the attention mechanism and decoder. The non-linear mappings $(\mathcal{F}_{h,1}, \ldots, \mathcal{F}_{h,T})$ and $\mathcal{F}_y$ are unchanged between the original and perturbed models. Hence, the hidden layer activation for the t-th word in the perturbed model $\tilde{\boldsymbol{h}}_t$ and the associated prediction of the i-th output $\tilde{y}_i$ are given by:

$$\tilde{\boldsymbol{h}}_t = \boldsymbol{h}_t + \Delta\boldsymbol{h}_t; \quad \tilde{y}_i = \mathcal{F}_y\left(\tilde{\boldsymbol{h}}_{1:T}, \tilde{\boldsymbol{y}}_{<i}; \boldsymbol{\theta}\right). \tag{3.2}$$

The aim of the project is to uncover perturbations $\Delta\boldsymbol{h}_t$ that enable the network's perturbed predictions $\tilde{\boldsymbol{y}}_{1:M}$ to be controlled in a predictable way. The nature of the perturbations is specific to the task for which the model is to be controlled. For instance, controlling a GEC model to generate fewer or more grammatical error corrections belonging to a particular part-of-speech (PoS), such as nouns, determiners, verbs, prepositions, etc., requires perturbations

$\Delta \boldsymbol{h}_t$ specific to that PoS. Likewise, correcting for gender bias requires perturbations $\Delta \boldsymbol{h}_t$ which correspond to gender features.

The simplest form of network perturbation is one in which individual nodes in the hidden layer are stimulated. Here, stimulation refers to perturbing the hidden layer activation of the i-th node by its activation $\boldsymbol{h}_t[i]$ multiplied by a scalar factor α:

$$\Delta \boldsymbol{h}_t[i] = \alpha \boldsymbol{h}_t[i]. \tag{3.3}$$

Though this is a relatively trivial means of perturbing a network, it is highly interpretable, as it enables analysis of the contribution of individual neurons to GEC models. With that, such questions can be asked as:

1. How important are individual nodes in the hidden layer to the final output of the GEC model?

2. Are there individual nodes which are responsible for particular aspects of grammatical error correction?

3. Can the output of a GEC model be controlled through perturbation of individual nodes?

Inspired by work in the field of computer vision [25, 38], stimulation of individual neurons has been the predominant means of network perturbation in the recent literature surrounding NMT [2, 8, 10]. These works have primarily focused on answering the first of the three proposed questions through neuron ablation studies; the importance of neurons within systems is evaluated by masking their activations' and measuring the impact on translation quality. This project aims to extend these works by addressing the latter two more interesting, but more challenging questions in the context of GEC. The following section outlines a methodology for finding neurons deemed important in a GEC model.

3.1 Ensemble Correlation Analysis

Ensemble correlation analyses investigate the extent to which neural networks exhibit *convergent learning* [22], a phenomenon in which individual representations learned by multiple networks trained on the same data set and same task converge to a set of features which are either individually similar between networks or where subsets of features span similar low-dimensional spaces. There are many ways in which an ensemble of models can be generated. The multiple networks may be models from different training epochs, trained with different random initialisations (random seeds), or even different architectures- all realistic

scenarios. In this project, the topology of network $\mathcal{M}$ was restricted to the RNN-based architecture presented in Chapter 2. An ensemble of M different models was generated through use of different random initialisations of the network parameters. For the given topology, the network parameters $\boldsymbol{\theta}$ were randomly initialised through M separate draws of the prior distribution over the model parameters $p(\boldsymbol{\theta} \mid \mathcal{M})$:

$$\tilde{\boldsymbol{\theta}}^{(i)} \sim p(\boldsymbol{\theta} \mid \mathcal{M}). \tag{3.4}$$

Each $\tilde{\boldsymbol{\theta}}^{(i)}$ was then used as a separate model initialisation, and trained to give the Maximum-Likelihood Estimate (MLE) of parameters $\tilde{\boldsymbol{\theta}}^{(i)}$ using the same supervised data set and training criteria. Thus, each $\boldsymbol{\theta}^{(i)}$ can then be viewed as a 'sample' of the model parameters at a 'local optimum'. The set of M trained model parameters yields the ensemble $\mathcal{E} = \left\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(M)}\right\}$. Due to the random initialisations, the representations of the model parameters are completely distributed: elements of one model cannot directly be related to another's. Consequently, general ensemble correlation approaches are required to uncover learned representations shared between models.

Crucial to setting up the mathematical details of an ensemble correlation analysis is the definition of the representation of a neuron. In particular, the interest is not the neuron's response to random data, but rather the representation of features corresponding to a relevant data set (e.g. written learner text). A neuron's representation can, therefore, be constructed as its set of responses over a finite set of inputs, such as those drawn from the training or validation set. This lookup table of `network input` $\rightarrow$ `neuron output` mappings gives a portrayal of that neuron's functional form.

More concretely, given an ensemble of M different models and a data set $\mathcal{D} = \{\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:K}\}$, let $\boldsymbol{h}_t^m[i]$ denote the hidden layer activation of the i-th node in the encoder output layer of the m-th model for the t-th input word w_t. The i-th neuron of the m-th model is then represented by the random vector $\boldsymbol{x}_i^m$ which maps each input word to it's corresponding neuron activation:

$$\boldsymbol{x}_i^m : t \mapsto \boldsymbol{h}_t^m[i]. \tag{3.5}$$

It is worth noting that this vector, a single neuron's response over the entire data set, differs from the frequently-considered vector of the neuron representations over the hidden layer for a single input. With this view, a neuron is represented as a single vector in the high-dimensional space $\mathbb{R}^T$. Extending this view from an individual neuron to the set of D neurons in a hidden layer, the hidden layer $\boldsymbol{x}^m$ is defined by the subspace spanned by its neurons i.e., a subspace in $\mathbb{R}^T \times \mathbb{R}^D$:

$$\boldsymbol{x}^m = \{\boldsymbol{x}_1^m, \ldots, \boldsymbol{x}_D^m\}. \tag{3.6}$$

With these definitions, the hidden layer representations over pairs of models $\boldsymbol{x}^{m_1}$ and $\boldsymbol{x}^{m_2}$ of the same architecture but different weight initialisations can be analysed.

Due to symmetries in the topology and weight initialisation procedures, for any given vector of weights that is found, one could create many equivalent solutions simply by permuting the orders of the nodes within the hidden layer and permuting the outgoing weights accordingly. The question this raises is that if allowed to permute the nodes of the first network, to what extent can it be brought into alignment with the other? To do so requires an appropriate measure of similarity between neurons, such that equivalent or nearly-equivalent neurons across networks can be found. It is hypothesised that neurons that align strongly with the neurons of other networks have learned largely the same feature representations, meaning similar important neurons emerge in different models.

The simplest form of ensemble correlation analysis, *Maximum* and *Minimum Ensemble Correlation (Max* and *Min Corr)* [2] methods involve correlating the learned representations of individual neurons on a one-to-one basis. The metric for correlation is the *Pearson correlation coefficient (PCC)*. Given a pair of mappings for different models and different neurons $\left(\boldsymbol{x}_i^{m_1}, \boldsymbol{x}_j^{m_2}\right)$, the Pearson correlation coefficient is defined as:

$$\rho\left(\boldsymbol{x}_i^{m_1}, \boldsymbol{x}_j^{m_2}\right) = \frac{\sum_{t=1}^{T}\left(\boldsymbol{h}_t^{m_1}[i] - \bar{\boldsymbol{h}}^{m_1}[i]\right)\left(\boldsymbol{h}_t^{m_2}[j] - \bar{\boldsymbol{h}}^{m_2}[j]\right)}{\sqrt{\sum_{t=1}^{T}\left(\boldsymbol{h}_t^{m_1}[i] - \bar{\boldsymbol{h}}^{m_1}[i]\right)^2}\sqrt{\sum_{t=1}^{T}\left(\boldsymbol{h}_t^{m_2}[j] - \bar{\boldsymbol{h}}^{m_2}[j]\right)^2}}, \tag{3.7}$$

where $\bar{\boldsymbol{h}}^{m_1}[i]$ is the mean of the activations for the i-th neuron in the m-th model over all words t:

$$\bar{\boldsymbol{h}}^{m_1}[i] = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{h}_t^{m_1}[i], \tag{3.8}$$

and analogously for $\bar{\boldsymbol{h}}^{m_2}[j]$. PCC is used as it is independent of the scale of the activations of neurons. The Max and Min Corr algorithms outline two standard ways in which the neurons within each model are ranked according to their PCC scores.

For the i-th neuron in the m-th model, the Max Corr algorithm seeks to find neuron with the highest PCC from all other models in the ensemble:

$$\text{Max Corr}\left(\boldsymbol{x}_i^m\right) = \max_{m' \neq m, j}\left|\rho\left(\boldsymbol{x}_i^m, \boldsymbol{x}_j^{m'}\right)\right|. \tag{3.9}$$

Neurons in the m-th model are then ranked according to their Max Corr score, revealing individual neurons which capture features strongly between two models.

Closely related to the Max Corr algorithm is the Min Corr algorithm. Min Corr operates by finding neurons most correlated with $\boldsymbol{x}_i^m$ amongst each of the other models in the ensemble

Figure 3.1 An illustration of the two correlation ranking methods, showing how to compute the score for one neuron using each of the methods. Here the number of models is $M = 3$, each having four neurons. The Pearson correlation coefficients are computed over time.

and selecting the one with the lowest correlation:

$$\text{Min Corr}\left(\boldsymbol{x}_i^m\right) = \min_{m' \neq m} \max_j \left| \rho\left(\boldsymbol{x}_i^m, \boldsymbol{x}_j^{m'}\right) \right|. \tag{3.10}$$

Neurons in the m-th model are then ranked according to their Min Corr score, revealing individual neurons which are well correlated with many other models.

The nature of the data set governs the learned feature representations correlated neurons capture. Ensemble correlation can be performed in a completely *unsupervised* way: hidden layer activations can be correlated over an entire data set to give neurons important for the overall GEC model's operation. The natural progression from unsupervised correlation is asking whether or not it is possible to find individual nodes which are responsible for particular aspects of grammatical error correction. This can be achieved through annotation of the data set $\mathcal{D}$, in the form of class labels $\boldsymbol{l}_{1:T}$ for the input sequence $\boldsymbol{w}_{1:T}$. Such labels sets may be PoS tags, in order to find nodes responsible for particular parts of speech. Alternatively, they may be error correction annotations, in pursuance of discovering nodes responsible for particular types of grammatical error correction. Both are realistic scenarios in the context of GEC. This project focused on the former, generating PoS labels using the methodology described in Chapter 2.3.2. The labels $\boldsymbol{l}_{1:T}$ for the input sequence $\boldsymbol{w}_{1:T}$ are then used as proxy labels for the sequence of hidden layer activations $\boldsymbol{h}_{1:T}$.

For each label l, a separate sub-set of data is formed, consisting of all hidden layer activations $\boldsymbol{h}_t$ labelled as l. Correlating hidden layer activations over the sub-set of data for a particular label reveals neurons important for that feature in the GEC model's operation. In this, there is an underlying assumption that individual nodes influence particular aspects of the error correction system.

3.2 Impact of Node Perturbation

In order to verify that neurons ranked highly by these correlation algorithms are in fact important for the GEC model, the effect of erasing neurons is evaluated. The literature shows erasing individual hidden-units to be a powerful technique for analysing the operation of DNNs [21]. Formally, for the m-th model, let $\boldsymbol{W}_m$ denote the $D \times D$ permutation matrix returned by the ensemble correlation method which permutes the neurons in the hidden layer activation vector $\boldsymbol{h}_t^m$, such that they are arranged from greatest to least importance:

$$\hat{\boldsymbol{h}}_t^m = \boldsymbol{W}_m^T \boldsymbol{h}_t^m. \tag{3.11}$$

The top k neurons are erased by multiplying the ordered activation vector $\hat{\boldsymbol{h}}_t^m$ by a diagonal matrix $\boldsymbol{\Phi}$, where the first k diagonal elements of $\boldsymbol{\Phi}$ are zeros, in order to erase the top k neurons, and the remaining elements ones, leaving the remaining $(D-k)$ neurons unchanged:

$$\check{\boldsymbol{h}}_t^m = \boldsymbol{\Phi} \boldsymbol{W}_m^T \boldsymbol{h}_t^m. \tag{3.12}$$

The neurons are then arranged back into their original order by undoing the transformation performed by the permutation matrix:

$$\tilde{\boldsymbol{h}}_t^m = \boldsymbol{W}_m \boldsymbol{\Phi} \boldsymbol{W}_m^T \boldsymbol{h}_t^m. \tag{3.13}$$

The full procedure for neuron erasure is shown diagrammatically in Figure 3.2. The impact of erasing individual neurons in the hidden layer on the performance of the GEC model is quantified by computing the GLEU score of the perturbed model and comparing it to the GLEU score of the baseline model. It is reasoned that erasing neurons from the top of the ranked list should impact translation performance to a greater degree than erasing neurons from the bottom, as the higher ranked neurons are of greater importance to the model.

Whilst neuron erasure is an effective tool for assessing the contribution of individual neurons to a network, it does not reveal how modifying neuron activations can affect model performance. The notion of neuron erasure can be extended to stimulation of neuron activations: rather than clamping the activations' of certain neurons to zero, the activations are perturbed by a scalar factor α:

$$\tilde{\boldsymbol{h}}_t[i] = \boldsymbol{h}_t[i] + \alpha \boldsymbol{h}_t[i]. \tag{3.14}$$

Factors of $\alpha > 0$ correspond to over-stimulation and $\alpha < 0$ under-stimulation. Neuron erasure is recovered for $\alpha = -1$, as $\tilde{\boldsymbol{h}}_t[i] = 0$. The impact of stimulating individual neurons

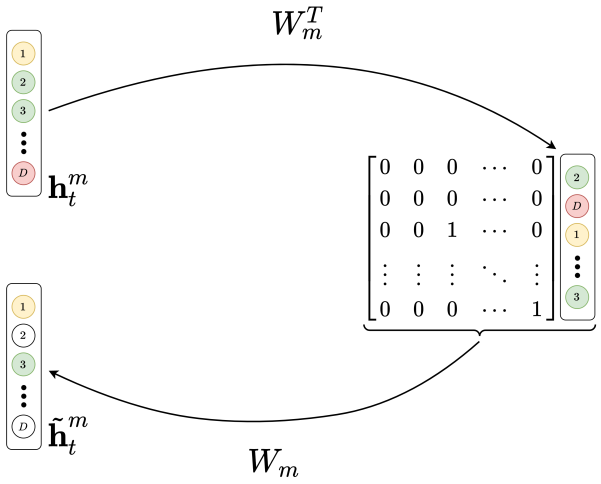


Figure 3.2 Network node erasure. The hidden layer activation vector is permuted by $\boldsymbol{W}_m$, such that the nodes are ordered from highest to lowest importance. The top two elements of the order activation vector are set to zero, through pre-multiplication by a diagonal matrix. The neurons are rearranged back into their original order by inverting the operation of the permutation matrix.

in the hidden layer on the performance of the GEC model can be evaluated in much the same way as neuron erasure, through computation of the GLEU score of the perturbed model and comparing it to the GLEU score of the baseline model.

A similar procedure can be followed for challenging the importance of neurons ranked by the permutation matrix $\boldsymbol{W}_{m,l}$ returned by an ensemble correlation analysis for the PoS with label l. It is hypothesised that erasing neurons from the top of the ranked list for a given PoS should impact the GEC model's predictions for that PoS to a far greater degree than all other parts of speech, thus yielding PoS control.

Controlling the GEC model for particular aspects of grammatical error correction is an unconventional NMT task. Hence, bespoke metrics for quantifying the degree of network control are required. The output of the perturbed GEC model $\tilde{\boldsymbol{y}}_{1:M}$ can be compared to the output of the baseline (unperturbed) model $\hat{\boldsymbol{y}}_{1:L}$ in a number of different ways. Perhaps the most intuitive metric for comparison is computation of the relative number of grammatical errors corrected between the two models. First, ERRANT is run to assess the output translations against the input sequence $\boldsymbol{w}_{1:T}$ to reveal the frequency of grammatical edits:

$$\hat{N} = ERRANT(\boldsymbol{w}_{1:T}, \hat{\boldsymbol{y}}_{1:L}); \quad \tilde{N} = ERRANT(\boldsymbol{w}_{1:T}, \tilde{\boldsymbol{y}}_{1:M}). \tag{3.15}$$

The relative change in the number of grammatical error corrections between the original and perturbed translations is then:

$$\text{Rel. change } = \frac{\tilde{N} - \hat{N}}{\hat{N}}. \tag{3.16}$$

A controllable scheme is easily identifiable as one for which the relative change in the number of grammatical error corrections can be increased for specific parts of speech and not others.

One shortcoming in using the relative PoS error metric for model comparison is the effect of the binary outcome at the decoder: there is either a PoS error or no PoS error. If using a free-running decoding scheme with a greedy search, when this binary outcome flips, the back-history of the predicted sequence is changed, an event described in detail in Chapter 2.2. This results in a cascade effect of errors/no-errors in the translation. The system is made robust to model history errors by decoding using teacher-forcing.

The binary outcome at the decoder arises due to the model making 'hard' decisions in its prediction of the output sequence: the i-th predicted output symbol $\hat{y}_i$ is sampled from the posterior probability mass function over the vocabulary. Rather than working with the symbols of the output translations, a metric can be formed which measures the dissimilarity between the probability mass functions of the perturbed and unperturbed models. This can be achieved through use of the *Kullback-Leibler (KL) divergence* between the two distributions. The aim is still to be able to assess the influence of perturbing the model on the encoder side on the predictions for a particular PoS. Thus, PoS tags $\boldsymbol{l}_{1:K}$ are assigned to the target sequence $\boldsymbol{y}_{1:K}$. The system's predictions are then grouped by target PoS, and the KL-divergence averaged over all target words with the class label l of interest:

$$\bar{D}_l = \frac{\sum\limits_{i:l_i=l}^{M} \boldsymbol{KL}\left(P(y_i \mid \boldsymbol{h}_{1:T}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}) \| P(y_i \mid \tilde{\boldsymbol{h}}_{1:T}, \boldsymbol{y}_{<i}; \boldsymbol{\theta})\right)}{\sum\limits_{i:l_i=l}^{M} 1}, \tag{3.17}$$

where M is the minimum sequence length of the perturbed and unperturbed translations, as there is no probability mass function generated beyond the end of the predicted sequence. Note that since decoding is performed using teacher forcing, the ground truth sequence $\boldsymbol{y}_{<i}$ is used in the prediction of the i-th output. $\bar{D}_l$ quantifies the average extent to which the probability mass function of the perturbed model is shifted relative to the baseline model for a target PoS. Hence, it is an assessment of how much the model's predictions change for a given PoS. In a similar way to the relative error metric, a controllable scheme is identifiable

as one for which the average KL-divergence can be increased for particular parts of speech and not others.

Chapter 4

Network Direction Perturbation

Network node perturbation described in the previous chapter inherently assumes that individual nodes or groups of nodes can operate in isolation. However, deep learning approaches typically use highly distributed representations, combining impacts from many nodes together to yield hidden layer activations. The results presented in Chapter 5.3 verify this; they demonstrate that network node perturbation is quite limited for tasks like GEC. Clearly, this motivates extending the scope of network perturbation from treating individual nodes to perturbing in network *directions*, or linear combinations of nodes. This yields the same form of network perturbation described before:

$$\tilde{\boldsymbol{h}}_t = \boldsymbol{h}_t + \Delta\boldsymbol{h}_t, \tag{4.1}$$

except now the perturbation $\Delta\boldsymbol{h}_t$ is in a general direction instead of being particular nodes.

4.1 Gradient-Based Approach

In constructing the global framework for network perturbation, the system's operation was decomposed into two non-linear mappings. The first was a mapping of the T-length sequence of input words $\boldsymbol{w}_{1:T}$ to a T-length sequence of hidden layer activations $\boldsymbol{h}_{1:T}$. The second was a mapping from the sequence of hidden layer activations and the sequence of $i-1$ previous targets $\boldsymbol{y}_{<i}$ (in teacher-forcing) to the network's prediction of the i-th output $\hat{y}_i$:

$$\boldsymbol{h}_t = \mathcal{F}_{h,t}\left(\boldsymbol{w}_{1:T};\boldsymbol{\theta}\right); \quad \hat{y}_i = \mathcal{F}_y\left(\boldsymbol{h}_{1:T},\boldsymbol{y}_{<i};\boldsymbol{\theta}\right). \tag{4.2}$$

Rather than mapping to the prediction for the i-th output, the second non-linear transformation can be generalised as a mapping from the sequence of hidden layer activations to a general

scalar-valued prediction $f\left(\boldsymbol{h}_{1:T}\right)$:

$$\boldsymbol{h}_t = \mathcal{F}_{h,t}\left(\boldsymbol{w}_{1:T}; \boldsymbol{\theta}\right); \quad f\left(\boldsymbol{h}_{1:T}\right) = \mathcal{F}_f\left(\boldsymbol{h}_{1:T}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}\right). \tag{4.3}$$

This scalar-valued prediction may be one of many scalar or vector-valued outputs generated by the model, such as a single element of a predicted probability mass function. For instance, it could be the network's prediction for the target word, a prediction for a particular PoS, or the prediction for a grammatical error being corrected, all realistic scenarios in the context of GEC. The most logical starting point for network direction perturbation is finding the direction $\Delta\boldsymbol{h}_t$ which maximises the change in the network's predicted output. This direction can be formalised mathematically as the gradient of the scalar-valued prediction with respect to the hidden layer activation at time t:

$$\Delta\boldsymbol{h}_t = \frac{\alpha}{Z} \left.\frac{\partial f\left(\boldsymbol{h}_{1:T}\right)}{\partial \boldsymbol{h}_t}\right|_{\boldsymbol{h}_{1:T}}, \tag{4.4}$$

where the normalising constant Z enforces the vector of derivatives to be unit magnitude in the l_2-norm:

$$Z = \left\| \left.\frac{\partial f\left(\boldsymbol{h}_{1:T}\right)}{\partial \boldsymbol{h}_t}\right|_{\boldsymbol{h}_{1:T}} \right\|_2. \tag{4.5}$$

Normalising the derivative vector to be unit magnitude enables the degree of perturbation to be set entirely by the scalar-valued factor α under the framework for network perturbation.

The results presented in Chapter 5.3 demonstrate that perturbing the hidden layer activations in the gradient directions is an effective way of modifying the network's predictions. This raises the question of whether the network naturally arranges node directions which influence particular parts of speech or particular types of grammatical error correction. Should such directions exist, the network could be perturbed in the direction corresponding to a particular PoS or particular type of error correction in order to control the network's predictions for that attribute in a predictable and reliable way. The gradient-based approach can, therefore, be viewed as motivation for the subsequent direction focused methods, which seek to find network node directions which are responsible for specific parts of speech.

4.2 Singular Vector Canonical Correlation Analysis

The assumption in Chapter 3 that individual nodes can be turned on and off to yield a controllable system is quite limited for the task of GEC. This is highlighted by the results exhibited in Chapter 5.3, which demonstrate that network node perturbation does not give

the level of control required for this project. This could either be because the neuron representations are unique (i.e. not convergent), or because the best possible one-to-one mapping is insufficient to fully describe how one neuron representation is related to another. A one-to-one mapping is simply a mapping of the activations of one network into the activations of another through a permutation matrix- a square matrix constrained such that each row and each column contains a single one and the remaining entries zeros. This implies that the second network's representation space is the same as the first's. However, in general this will not be the case, and there will be a transformation which relates the two representation spaces. Relaxing the constraints on the permutation matrix, a general projection matrix can be identified which characterises the extent to which the second network's representation space is an affine transformation of the first's. This leads to a strict generalisation of the ensemble correlation methods described in the previous chapter: correlating learned representations at the individual neuron level reveals salient neurons, whereas correlating linear combinations of neurons uncovers important directions.

Singular Vector Canonical Correlation Analysis (SVCCA) is a recent method for analysing neural networks, developed by Raghu et al. [29]. The name SVCCA is an amalgamation of Singular Value Decomposition (SVD) and Canonical Correlation Analysis (CCA) [11], the two fundamental operations of which it comprises. Based on an analysis of each neuron's activation vector - the scalar outputs it emits on input data points - it acts as tool for quickly comparing two representations in a way that is both invariant to affine transformations (allowing comparison between different networks and layers) and fast to compute (allowing more comparisons to be calculated than with previous methods).

4.2.1 Singular Value Decomposition

The first stage in SVCCA is a Singular Value Decomposition (SVD) of each of the subspaces $\boldsymbol{x}^{m_1}$, $\boldsymbol{x}^{m_2}$. The SVD is a factorisation of a real or complex matrix that generalizes the eigendecomposition of a square normal matrix to any $T \times D$ matrix. Specifically, the SVD of the real $T \times D$ matrix $\boldsymbol{x}^m$ is:

$$\boldsymbol{x}^m = \boldsymbol{U}^m \boldsymbol{\Sigma}^m \left(\boldsymbol{V}^m\right)^T, \tag{4.6}$$

where $\boldsymbol{U}^m \in \mathbb{R}^{T \times T}$ is an orthogonal matrix, $\boldsymbol{\Sigma}^m \in \mathbb{R}^{T \times D}$ is a diagonal matrix, with diagonal entries σ_i^m (the 'singular values') sorted such that $\sigma_1^m \geq \sigma_2^m \geq \ldots \geq \sigma_p^m \geq 0$, where $p = \min(T, D)$, and $\boldsymbol{V}^m \in \mathbb{R}^{D \times D}$ is an orthogonal matrix. The columns of $\boldsymbol{U}^m$ and the columns of $\boldsymbol{V}^m$ are the (normalised) eigenvectors of $\boldsymbol{x}^m \left(\boldsymbol{x}^m\right)^T$ and $\left(\boldsymbol{x}^m\right)^T \boldsymbol{x}^m$ respectively, and are often termed the left-singular vectors and right-singular vectors of $\boldsymbol{x}^m$. The diagonal

entries of $\mathbf{\Sigma}^m$ are the square roots of the eigenvalues of $\boldsymbol{x}^m \left(\boldsymbol{x}^m\right)^T$, which are the same as the eigenvalues of $\left(\boldsymbol{x}^m\right)^T \boldsymbol{x}^m$.

Of the D left-singular vectors $\{\hat{\boldsymbol{x}}_1^m, \ldots, \hat{\boldsymbol{x}}_D^m\}$ with associated singular values $\{\sigma_1^m, \ldots, \sigma_D^m\}$, the top D_m vectors are retained, where D_m is the smallest value that:

$$\sum_{i=1}^{D_m} |\sigma_i^m| \left(\geq 0.99 \sum_{i=1}^{D} |\sigma_i^m| \right). \tag{4.7}$$

That is, 99% of the variation of $\boldsymbol{x}^m$ is explainable by the top D_m vectors. Hence, the subspaces:

$$\hat{\boldsymbol{x}}^{m_1} = \{\hat{\boldsymbol{x}}_1^{m_1}, \ldots, \hat{\boldsymbol{x}}_{D_1}^{m_1}\}; \quad \hat{\boldsymbol{x}}^{m_2} = \{\hat{\boldsymbol{x}}_1^{m_2}, \ldots, \hat{\boldsymbol{x}}_{D_2}^{m_2}\} \tag{4.8}$$

comprise of the most important directions of the original subspaces $\boldsymbol{x}^{m_1}, \boldsymbol{x}^{m_2}$. The motivation for this step is that many low variance directions are primarily noise [29]. Removing these directions improves the performance of the subsequent CCA, which suffers shortfalls in determining how many directions are important to the original space, which is exactly the strength of SVD.

4.2.2 Canonical Correlation Analysis

Correlation analysis is a statistical technique used extensively to evaluate the strength of relationship between two multidimensional variables. It seeks a pair of linear transformations, one for each of the sets of variables, such that when the set of variables are transformed the corresponding co-ordinates are maximally correlated.

A new coordinate system for $\hat{\boldsymbol{x}}_i^{m_1}$ is defined by choosing a direction $\boldsymbol{w}_1$ and projecting $\hat{\boldsymbol{x}}_i^{m_1}$ onto that direction: $\hat{\boldsymbol{x}}^{m_1} \rightarrow \langle \boldsymbol{w}_1, \hat{\boldsymbol{x}}^{m_1} \rangle$, where $\langle \boldsymbol{w}_1, \hat{\boldsymbol{x}}^{m_1} \rangle$ denotes the Euclidean inner product of the vectors $\boldsymbol{w}_1$ and $\hat{\boldsymbol{x}}^{m_1}$. Doing the same for $\hat{\boldsymbol{x}}_i^{m_2}$ by choosing a direction $\boldsymbol{w}_2$ yields the transformed subspace: $\hat{\boldsymbol{x}}^{m_2} \rightarrow \langle \boldsymbol{w}_2, \hat{\boldsymbol{x}}^{m_2} \rangle$.

The objective of CCA is to find the linear transformations $\boldsymbol{w}_1$, $\boldsymbol{w}_2$ which maximise the PCCs between the two transformed vectors $\tilde{\boldsymbol{x}}^{m_1}$, $\tilde{\boldsymbol{x}}^{m_2}$. Assuming $\hat{\boldsymbol{x}}^{m_1}$ and $\hat{\boldsymbol{x}}^{m_2}$ to be zero-mean, the problem can be formulated as:

$$\boldsymbol{w}_1, \boldsymbol{w}_2 = \underset{\boldsymbol{w}_1, \boldsymbol{w}_2}{\arg\max}\, \rho(\hat{\boldsymbol{x}}^{m_1}\boldsymbol{w}_1, \hat{\boldsymbol{x}}^{m_2}\boldsymbol{w}_2) \tag{4.9}$$

$$= \underset{\boldsymbol{w}_1, \boldsymbol{w}_2}{\arg\max} \frac{\langle \hat{\boldsymbol{x}}^{m_1}\boldsymbol{w}_1, \hat{\boldsymbol{x}}^{m_2}\boldsymbol{w}_2 \rangle}{\|\hat{\boldsymbol{x}}^{m_1}\boldsymbol{w}_1\| \cdot \|\hat{\boldsymbol{x}}^{m_2}\boldsymbol{w}_2\|}. \tag{4.10}$$

For the case where $\boldsymbol{x}^{m_1}$ or $\boldsymbol{x}^{m_2}$ (and hence $\hat{\boldsymbol{x}}^{m_1}$ or $\hat{\boldsymbol{x}}^{m_2}$) are not zero-mean, the random vectors are centred on their first moment's prior to computing the correlation. Denoting the

empirical expectation of the variable $\boldsymbol{x}$ by $\hat{\mathbb{E}}[\boldsymbol{x}]$, the correlation expression can be rewritten in the following form:

$$\boldsymbol{w}_1, \boldsymbol{w}_2 = \underset{\boldsymbol{w}_1, \boldsymbol{w}_2}{\arg\max} \frac{\hat{\mathbb{E}}\left[\langle \boldsymbol{w}_1, \hat{\boldsymbol{x}}^{m_1} \rangle \langle \boldsymbol{w}_2, \hat{\boldsymbol{x}}^{m_2} \rangle\right]}{\sqrt{\hat{\mathbb{E}}\left[\langle \boldsymbol{w}_1, \hat{\boldsymbol{x}}^{m_1} \rangle^2\right] \hat{\mathbb{E}}\left[\langle \boldsymbol{w}_2, \hat{\boldsymbol{x}}^{m_2} \rangle^2\right]}} \tag{4.11}$$

$$= \underset{\boldsymbol{w}_1, \boldsymbol{w}_2}{\arg\max} \frac{\boldsymbol{w}_1^T \hat{\mathbb{E}}\left[\hat{\boldsymbol{x}}^{m_1} \left(\hat{\boldsymbol{x}}^{m_2}\right)^T\right] \boldsymbol{w}_2}{\sqrt{\boldsymbol{w}_1^T \hat{\mathbb{E}}\left[\hat{\boldsymbol{x}}^{m_1} \left(\hat{\boldsymbol{x}}^{m_1}\right)^T\right] \boldsymbol{w}_1 \boldsymbol{w}_2^T \hat{\mathbb{E}}\left[\hat{\boldsymbol{x}}^{m_2} \left(\hat{\boldsymbol{x}}^{m_2}\right)^T\right] \boldsymbol{w}_2}}. \tag{4.12}$$

For the zero mean case ($\hat{\mathbb{E}}[\boldsymbol{x}^{m_1}] = \hat{\mathbb{E}}[\boldsymbol{x}^{m_2}] = \mathbf{0}$), the sample covariance matrix of $(\hat{\boldsymbol{x}}^{m_1}, \hat{\boldsymbol{x}}^{m_2})$ is:

$$\hat{\mathbb{E}}\left[\begin{pmatrix} \hat{\boldsymbol{x}}^{m_1} \\ \hat{\boldsymbol{x}}^{m_2} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{x}}^{m_1} \\ \hat{\boldsymbol{x}}^{m_2} \end{pmatrix}^T\right] = \hat{\mathbb{E}}\begin{bmatrix} \hat{\boldsymbol{x}}^{m_1}(\hat{\boldsymbol{x}}^{m_1})^T & \hat{\boldsymbol{x}}^{m_1}(\hat{\boldsymbol{x}}^{m_2})^T \\ \hat{\boldsymbol{x}}^{m_2}(\hat{\boldsymbol{x}}^{m_1})^T & \hat{\boldsymbol{x}}^{m_2}(\hat{\boldsymbol{x}}^{m_1})^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}, \tag{4.13}$$

where $\boldsymbol{\Sigma}_{11}, \boldsymbol{\Sigma}_{22}$ are the covariance and $\boldsymbol{\Sigma}_{12}, \boldsymbol{\Sigma}_{21}$ cross-covariance terms respectively, with $\boldsymbol{\Sigma}_{12} = \boldsymbol{\Sigma}_{21}^T$. Hence, eq. (4.12) can be rewritten as:

$$\boldsymbol{w}_1, \boldsymbol{w}_2 = \underset{\boldsymbol{w}_1, \boldsymbol{w}_2}{\arg\max} \frac{\boldsymbol{w}_1^T \boldsymbol{\Sigma}_{12} \boldsymbol{w}_2}{\sqrt{\boldsymbol{w}_1^T \boldsymbol{\Sigma}_{11} \boldsymbol{w}_1 \boldsymbol{w}_2^T \boldsymbol{\Sigma}_{22} \boldsymbol{w}_2}}. \tag{4.14}$$

Since the solution of eq. (4.14) is invariant with respect to length scaling of $\boldsymbol{w}_1$ and also $\boldsymbol{w}_2$, either together or independently, the CCA optimisation problem formulated in eq. (4.14) is equivalent to maximising the equivalent objective f:

$$f(\boldsymbol{w}_1, \boldsymbol{w}_2) = \boldsymbol{w}_1^T \boldsymbol{\Sigma}_{12} \boldsymbol{w}_2, \tag{4.15}$$

subject to the constraints $\boldsymbol{w}_1^T \boldsymbol{\Sigma}_{11} \boldsymbol{w}_1 = 1$ and $\boldsymbol{w}_2^T \boldsymbol{\Sigma}_{22} \boldsymbol{w}_2 = 1$. To find the optimal projections $\boldsymbol{w}_1, \boldsymbol{w}_2$, under the constraints, the Lagrangian $\mathcal{L}$ must be used:

$$\mathcal{L}(\boldsymbol{w}_1, \boldsymbol{w}_2, \lambda_1, \lambda_2) = \boldsymbol{w}_1^T \boldsymbol{\Sigma}_{12} \boldsymbol{w}_2 - \frac{\lambda_1}{2}\left(\boldsymbol{w}_1^T \boldsymbol{\Sigma}_{11} \boldsymbol{w}_1 - 1\right) - \frac{\lambda_2}{2}\left(\boldsymbol{w}_2^T \boldsymbol{\Sigma}_{22} \boldsymbol{w}_2 - 1\right). \tag{4.16}$$

Taking derivatives with respect to $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ and applying the zero derivative criteria:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_1} = \boldsymbol{\Sigma}_{12}\boldsymbol{w}_2 - \lambda_1 \boldsymbol{\Sigma}_{11}\boldsymbol{w}_1 = 0, \tag{4.17}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_2} = \boldsymbol{\Sigma}_{21}\boldsymbol{w}_1 - \lambda_2 \boldsymbol{\Sigma}_{22}\boldsymbol{w}_2 = 0. \tag{4.18}$$

Subtracting $\boldsymbol{w}_2^T$ times the second equation from $\boldsymbol{w}_1^T$ times the first gives:

$$0 = \boldsymbol{w}_1^T(\boldsymbol{\Sigma}_{12}\boldsymbol{w}_2 - \lambda_1 \boldsymbol{\Sigma}_{11}\boldsymbol{w}_1) - \boldsymbol{w}_2^T(\boldsymbol{\Sigma}_{21}\boldsymbol{w}_1 - \lambda_2 \boldsymbol{\Sigma}_{22}\boldsymbol{w}_2) \tag{4.19}$$

$$= \lambda_2 \boldsymbol{w}_2^T \boldsymbol{\Sigma}_{22}\boldsymbol{w}_2 - \lambda_1 \boldsymbol{w}_1^T \boldsymbol{\Sigma}_{11}\boldsymbol{w}_1, \tag{4.20}$$

which together with the constraints implies that $\lambda_1 = \lambda_2 = \lambda$. Assuming $\boldsymbol{\Sigma}_{22}$ is invertible, eq. (4.18) becomes:

$$\boldsymbol{w}_2 = \frac{\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{w}_1}{\lambda}. \tag{4.21}$$

Using this to eliminate $\boldsymbol{w}_2$ in equation eq. (4.17) gives:

$$\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{w}_1 = \lambda^2 \boldsymbol{\Sigma}_{11}\boldsymbol{w}_1. \tag{4.22}$$

which is a generalised eigenproblem of the form $A\boldsymbol{x} = \lambda B\boldsymbol{x}$ (albeit with λ^2 as the eigenvalue). Assuming that $\boldsymbol{\Sigma}_{11}$ is invertible, this can be equivalently written as:

$$\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{w}_1 = \lambda^2 \boldsymbol{w}_1, \tag{4.23}$$

which is a standard eigenproblem of the form $A\boldsymbol{x} = \lambda\boldsymbol{x}$. It is straightforward to show that the vector $\boldsymbol{w}_1$ can be found by first computing the SVD of:

$$\boldsymbol{\Sigma}_{11}^{-1/2}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1/2} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{4.24}$$

where the columns of the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ correspond to the sets of orthonormal left and right singular vectors respectively, and the singular values of matrix $\boldsymbol{S}$ correspond to the canonical correlations. By extracting the maximal left singular vector $\boldsymbol{u}_1$ of $\boldsymbol{U}$ (the first column on $\boldsymbol{U}$), then $\boldsymbol{w}_1$ is optimally $\boldsymbol{\Sigma}_{11}^{-1/2}\boldsymbol{u}_1$, and similarly, $\boldsymbol{w}_2$ is $\boldsymbol{\Sigma}_{22}^{-1/2}\boldsymbol{v}_1$, where $\boldsymbol{v}_1$ is the first column of the matrix $\boldsymbol{V}$. The extension to finding the matrices of K multiple directions $\boldsymbol{W}_1 = \left[\boldsymbol{w}_1^{(1)}, \ldots \boldsymbol{w}_1^{(K)}\right]$ and $\boldsymbol{W}_2 = \left[\boldsymbol{w}_2^{(1)}, \ldots \boldsymbol{w}_2^{(K)}\right]$ is clear: the corresponding first K

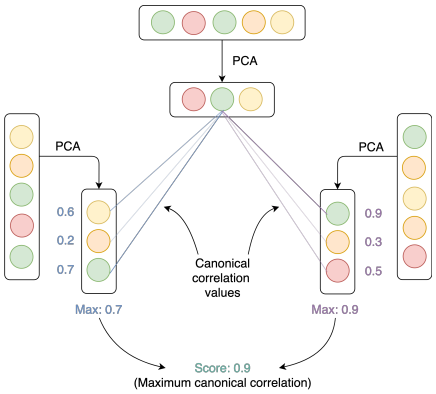


Figure 4.1 An illustration of the SVCCA direction ranking algorithm. The number of models is $M = 3$, each having five neurons. The canonical correlation values are computed over time.

singular vectors are taken accordingly. Doing so maximises the criterion:

$$\frac{\text{trace}\left(\boldsymbol{W}_1^T \boldsymbol{\Sigma}_{12} \boldsymbol{W}_2\right)}{\sqrt{\text{trace}\left(\boldsymbol{W}_1^T \boldsymbol{\Sigma}_{11} \boldsymbol{W}_1\right)} \sqrt{\text{trace}\left(\boldsymbol{W}_2^T \boldsymbol{\Sigma}_{22} \boldsymbol{W}_2\right)}}. \tag{4.25}$$

4.2.3 SVCCA Summary

1. Input: $\boldsymbol{x}^{m_1}, \boldsymbol{x}^{m_2}$
2. Perform: SVD$(\boldsymbol{x}^{m_1})$, SVD$(\boldsymbol{x}^{m_2})$. Output: $\hat{\boldsymbol{x}}^{m_1}, \hat{\boldsymbol{x}}^{m_2}$
3. Perform: CCA$(\hat{\boldsymbol{x}}^{m_1}, \hat{\boldsymbol{x}}^{m_2})$. Output: $\boldsymbol{W}_1, \boldsymbol{W}_2$

In a similar manner to stimulation of individual salient neurons in network node perturbation, the importance of SVCCA directions is challenged through perturbation of individual directions. Formally, for the m-th model, a complete SVCCA returns a $D \times D$ projection matrix $\boldsymbol{W}_m = \left[\boldsymbol{w}_m^{(1)}, \ldots \boldsymbol{w}_m^{(D)}\right]$. The D-dimensional hidden layer activation $\boldsymbol{h}_t^m$ is projected onto the space spanned by the columns of $\boldsymbol{W}_m$ through a change of basis:

$$\hat{\boldsymbol{h}}_t^m = \boldsymbol{W}_m^T \boldsymbol{h}_t^m. \tag{4.26}$$

Hence, the i-th element of $\hat{\boldsymbol{h}}_t^m$ is the projection of $\boldsymbol{h}_t^m$ onto the i-th SVCCA direction $\boldsymbol{w}_m^{(i)}$. Perturbations are applied in this transformed space by scaling individual elements of the D-dimensional vector $\hat{\boldsymbol{h}}_t^m$, in an entirely equivalent way to node stimulation in the original hidden layer activation space. The top k SVCCA directions are stimulated by pre-multiplying

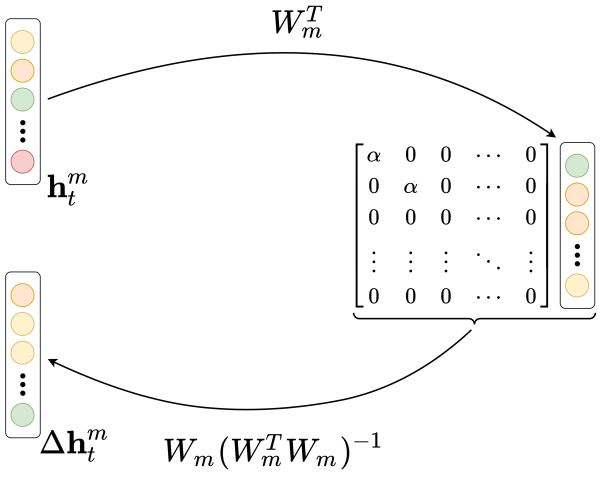


Figure 4.2 The activation output space is first transformed into another vector space by means of a projection matrix. Perturbations are then applied in this transformed space, in an entirely equivalent way to stimulating individual nodes in the original activation space. The perturbed vector is then transformed back into the original space by inverting the projection operation, thus returning the final perturbation to be applied.

$\hat{\boldsymbol{h}}_t^m$ by a diagonal matrix $\boldsymbol{\Phi}(\alpha)$:

$$\check{\boldsymbol{h}}_t^m = \boldsymbol{\Phi}(\alpha)\hat{\boldsymbol{h}}_t^m, \tag{4.27}$$

where:

$$\boldsymbol{\Phi}(\alpha)[i,j] = \begin{cases} \alpha & \text{if } i = j \text{ and } i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{4.28}$$

Hence, $\boldsymbol{\Phi}(\alpha)[i,i]$ dictates the degree of perturbation applied to the i-th element of $\hat{\boldsymbol{h}}_t^m$. Factors of $\alpha > 0$ correspond to over-stimulation and $\alpha < 0$ under-stimulation. Direction erasure is obtained for $\alpha = -1$, equivalent to node erasure in the original space. A second linear transformation is performed to bring the perturbed hidden layer activation $\check{\boldsymbol{h}}_t^m$ in the transformed space back into the original space spanned by $\boldsymbol{h}_t^m$, thus yielding the final perturbation:

$$\Delta\boldsymbol{h}_t^m = \boldsymbol{W}_m \left(\boldsymbol{W}_m^T \boldsymbol{W}_m\right)^{-1} \check{\boldsymbol{h}}_t^m = \boldsymbol{W}_m \check{\boldsymbol{h}}_t^m \tag{4.29}$$

$$= \boldsymbol{W}_m \boldsymbol{\Phi}(\alpha) \boldsymbol{W}_m^T \boldsymbol{h}_t^m, \tag{4.30}$$

where the first line follows from the fact that $\boldsymbol{W}_m$ is an orthonormal matrix, meaning $\boldsymbol{W}_m^T \boldsymbol{W}_m = \boldsymbol{I}$. It is clear that correlation-based methods are model specific; $\Delta\boldsymbol{h}_t^m$ is dependent on $\boldsymbol{W}_m$. For the case that $\boldsymbol{W}_m$ is a $D \times D$ permutation matrix, the perturbation

in eq. (4.30) collapses to stimulating the top k individual nodes in the original hidden layer activation space. Thus, the framework for network direction perturbation can be viewed as a strict generalisation of the mechanism outlined in the previous chapter for network node perturbation.

As with network node correlation, the unsupervised SVCCA procedure can be extended to find salient network directions for given parts of speech. The input sequence is first annotated with PoS class labels. Hidden layer activations are then correlated using SVCCA over the sub-set of hidden layer activation data for a particular PoS. This reveals directions in the activation space important for that PoS in the operation of the GEC model. The projection matrix returned $\boldsymbol{W}_{m,l}$ is then model specific and PoS dependent. The network direction perturbation $\Delta \boldsymbol{h}_{t,l}^{m}$ associated with the PoS with label l is found by scaling the first k columns of $\boldsymbol{W}_{m,l}$ by a scalar factor α_l. The overall perturbation $\Delta \boldsymbol{h}_{t}^{m}$ is obtained by summing the individual PoS perturbations over all PoS labels in the label set $\mathcal{P}$:

$$\Delta \boldsymbol{h}_{t}^{m} = \sum_{l \in \mathcal{P}} \Delta \boldsymbol{h}_{t,l}^{m} = \sum_{l \in \mathcal{P}} \boldsymbol{W}_{m,l} \boldsymbol{\Phi}(\alpha_l) \boldsymbol{W}_{m,l}^{T} \boldsymbol{h}_{t}^{m} \tag{4.31}$$

4.3 Logistic Classification

The aim of this section is to present an alternative, single model method for uncovering salient directions $\Delta \boldsymbol{h}_t$ to be used for model control in GEC systems. The task is set-up first by generating fully annotated (supervised) data specific to the control task. Each word w_t in the input sequence is assigned a label l_t from a label set $\mathcal{P}$, where the label set is specific to the control task of interest. The input set $\boldsymbol{w}_{1:T}$ is then fed into the network, and the set of hidden layer activations $\boldsymbol{h}_{1:T}$ at the encoder output layer recorded, giving a sequence of $\{w_t, l_t, \boldsymbol{h}_t\}$ trios. The problem can then be treated as a classification task, where the goal is to predict the label for the t-th word l_t from the D-dimensional hidden layer activation vector $\boldsymbol{h}_t$. The labels are taken to be disjoint, such that each hidden layer activation is assigned to a single label. This results in the input space being divided into *decision regions* $\mathcal{R}_l$, such that all hidden layer activations in $\mathcal{R}_l$ are assigned to the label l. The boundaries between decision regions are termed *decision boundaries* or *decision surfaces*. In this project, the form of classifier is restricted to linear models, meaning the decision boundaries are linear functions of the hidden layer activations $\boldsymbol{h}_t$, and thus defined by $(D-1)$-dimensional hyperplanes within the D dimensional input space. This restriction is based on the underlying assumption that if a classifier of simple linear architecture can predict the label assigned to a word, then the hidden layer representations implicitly encode this information.

The first stage in forming a posterior prediction for the discrete class label $l \in \mathcal{P}$ is a linear transformation of the hidden layer activations:

$$z_l = \boldsymbol{w}_l^T \boldsymbol{h}_t + b_l, \tag{4.32}$$

where $\boldsymbol{w}_l$ is a D-dimensional vector of weights and b_l a scalar-valued intercept term (or 'bias'). Let $\boldsymbol{\theta}^c$ denote the set of weights and biases over all classes, collectively referred to as the classifier parameters. In order to be a valid probability distribution, the posterior probabilities over the class labels must satisfy the sum-to-one and non-negativity constraints. The softmax function exponentiates and normalises z_l to obtain the desired posterior probability:

$$P(l|\boldsymbol{h}_t; \boldsymbol{\theta}^c) = \frac{\exp(z_l)}{\sum_{l' \in \mathcal{P}} \exp(z_{l'})} = \frac{\exp\left(\boldsymbol{w}_l^T \boldsymbol{h}_t + b_l\right)}{\sum_{l' \in \mathcal{P}} \exp\left(\boldsymbol{w}_{l'}^T \boldsymbol{h}_t + b_{l'}\right)}. \tag{4.33}$$

Bayes' decision rule states that a hidden layer activation $\boldsymbol{h}_t$ is assigned to the class label l with with the largest posterior probability: $P(l|\boldsymbol{h}_t; \boldsymbol{\theta}^c) > P(k|\boldsymbol{h}_t; \boldsymbol{\theta}^c)$ for all $k \neq l$. Thus, the decision boundary between classes l and k is given by the condition $P(l|\boldsymbol{h}_t; \boldsymbol{\theta}^c) = P(k|\boldsymbol{h}_t; \boldsymbol{\theta}^c)$. This corresponds to the $(D-1)$-dimensional hyperplane defined by:

$$\left(\boldsymbol{w}_l - \boldsymbol{w}_k\right)^T \boldsymbol{h}_t + b_l - b_k = 0. \tag{4.34}$$

Hence, the decision surfaces are linear functions of $\boldsymbol{h}_t$, despite the fact the exponential function is non-linear. Note, however, that due to the non-linear transformation performed by the softmax function, the model is not linear in the parameters.

The logistic classifier is trained on the $\{\boldsymbol{h}_t, l_t\}$ pairs to yield the maximum likelihood estimate (MLE) of classifier parameters $\boldsymbol{\theta}^c$:

$$\hat{\boldsymbol{\theta}}^c = \underset{\boldsymbol{\theta}^c}{\operatorname{argmax}} \left\{ \sum_{t=1}^{T} \log\left(P(l_t|\boldsymbol{h}_t; \boldsymbol{\theta}^c)\right) \right\} = \underset{\boldsymbol{\theta}^c}{\operatorname{argmax}} \left\{ \sum_{t=1}^{T} \log\left(\frac{\exp\left(\boldsymbol{w}_{l_t}^T \boldsymbol{h}_t + b_{l_t}\right)}{\sum_{l' \in \mathcal{P}} \exp\left(\boldsymbol{w}_{l'}^T \boldsymbol{h}_t + b_{l'}\right)} \right) \right\} \tag{4.35}$$

The full-form of the softmax probability in eq. (4.35) provides a clear interpretation for the learned weights: important directions in the logistic classifier are retained, whilst unimportant directions are diminished. The classifier's weights therefore provide a direct measure of the importance of each direction.

Inclusion of the bias term provides the classifier with greater flexibility in its learning of the decision boundaries which partition the hidden layer activation space. This enables the

logistic classifier to learn directions of potentially greater saliency, as the weight vectors $\boldsymbol{w}_l$ are not restricted to define hyperplanes which pass through the origin.

The perturbation directions $\Delta\boldsymbol{h}_{t,l}$ associated with the class label l can be formed in a number of ways from the learned classifier directions. Let $\hat{\boldsymbol{w}}_l$ denote the l_2-normalised classifier weights: $\hat{\boldsymbol{w}}_l = \boldsymbol{w}_l / \|\boldsymbol{w}_l\|_2$. In the simplest scheme, $\Delta\boldsymbol{h}_{t,l}$ is fixed to be $\hat{\boldsymbol{w}}_l$. The overall perturbation $\Delta\boldsymbol{h}_t$ is formed as weighted sum of all the class perturbations in the label set $\mathcal{P}$:

$$\Delta\boldsymbol{h}_t = \sum_{l \in \mathcal{P}} \alpha_l \Delta\boldsymbol{h}_{t,l} = \sum_{l \in \mathcal{P}} \alpha_l \hat{\boldsymbol{w}}_l \tag{4.36}$$

As the perturbations $\hat{\boldsymbol{w}}_l$ are of unit magnitude, the scalar factors α_l completely define the magnitude of the perturbations applied in the directions $\hat{\boldsymbol{w}}_l$. The interpretation of the scalar factors α_l is as follows. A value of $\alpha_l = 0$ produces no perturbation in the direction $\Delta\boldsymbol{h}_{t,l}$, the $\Delta\boldsymbol{h}_{t,l}$ term disappearing from the expression in eq. (4.36), whilst $\alpha_l < 0$ and $\alpha_l > 0$ correspond to under and over-stimulation of the direction $\Delta\boldsymbol{h}_{t,l} = \hat{\boldsymbol{w}}_l$ respectively. Since the perturbation $\Delta\boldsymbol{h}_t$ is independent of $\boldsymbol{h}_t$, the same perturbation is applied to the every hidden layer activation in the data set. In essence, this form of perturbation adds a fixed bias term to the hidden layer activations in the the l_2-normalised classifier directions.

In order to apply a different perturbation to each hidden layer activation vector, the perturbation directions $\Delta\boldsymbol{h}_{t,l}$ can be made a function of $\boldsymbol{h}_t$. One way in which this can be achieved is by scaling the directions $\hat{\boldsymbol{w}}_l$ by an amount given by the projection of $\boldsymbol{h}_t$ onto $\hat{\boldsymbol{w}}_l$:

$$\Delta\boldsymbol{h}_t = \sum_{l \in \mathcal{P}} \alpha_l \Delta\boldsymbol{h}_{t,l} = \sum_{l \in \mathcal{P}} \alpha_l \left(\hat{\boldsymbol{w}}_l^T \boldsymbol{h}_t\right) \hat{\boldsymbol{w}}_l \tag{4.37}$$

This form of perturbation for $\Delta\boldsymbol{h}_{t,l}$ has close ties with the framework for network node stimulation presented in Chapter 3.2. Rather than perturbing the hidden layer activations in the original activation space, they are projected onto the space spanned by the learned classifier directions. Perturbations are then applied in this transformed space, in an entirely equivalent way to node stimulation in the original activation space. Hence, the scalar factors α_l can be viewed as the level of network stimulation applied in the transformed space. The value $\alpha_l = -1$ is of particular significance; it results in complete erasure of the component of the hidden layer layer activation in the direction $\hat{\boldsymbol{w}}_l$, thus eliminating it completely from the perturbed vector $\tilde{\boldsymbol{h}}_t$. This is equivalent to ablating an individual node in the transformed vector space.

Multiple directions for each label $\boldsymbol{w}_l^{(1)}, \ldots, \boldsymbol{w}_l^{(K)}$ are obtained by recursively using the classifier, and eliminating used directions from the hidden layer activations after each iteration. Let $\boldsymbol{W}_l$ denote the $D \times K$ matrix who's columns are the l_2-normalised classifier weights for

the class label l: $\boldsymbol{W}_l = [\hat{\boldsymbol{w}}_l^{(1)}, \ldots, \hat{\boldsymbol{w}}_l^{(K)}]$. The projection of the hidden layer activation onto the first k classifier directions is then:

$$\Delta \boldsymbol{h}_t = \sum_{l \in \mathcal{P}} \alpha_l \Delta \boldsymbol{h}_{t,l} = \sum_{l \in \mathcal{P}} \alpha_l \sum_{i=1}^{k} \left(\hat{\boldsymbol{w}}_l^{(i)^T} \boldsymbol{h}_t \right) \hat{\boldsymbol{w}}_l^{(i)} = \sum_{l \in \mathcal{P}} \boldsymbol{W}_l (\boldsymbol{W}_l^T \boldsymbol{W}_l)^{-1} \boldsymbol{\Phi}(\alpha_l) \boldsymbol{W}_l^T \boldsymbol{h}_t \tag{4.38}$$

where the matrix $\boldsymbol{\Phi}(\alpha_l)$ is defined in eq. (4.28), scaling the top k directions by a scalar value α_l. This is in a equivalent form to the perturbations derived for the SVCCA PoS directions in eq. (4.31), demonstrating another symmetry between the methods presented.

Rather than hand-crafting the form of the perturbations using mathematical intuition, one can ask the precise question of what direction to perturb the hidden layer activations to maximise the change in the prediction of a particular label. This approach bears a close resemblance to the gradient-based methods discussed in Chapter 4.1. However, rather than taking the derivative of a scalar-valued prediction generated by the GEC model, the scalar-valued prediction is that given by the posterior probability mass function of the logistic classifier built in the activation function space. The direction can be formalised mathematically as the derivative of the softmax probability with respect to the hidden layer activation $\boldsymbol{h}_t$:

$$\frac{\partial P(l|\boldsymbol{h}_t; \boldsymbol{\theta}^c)}{\partial \boldsymbol{h}_t} = \frac{\exp\left(\boldsymbol{w}_l^T \boldsymbol{h}_t + b_l\right)}{\sum_{l' \in \mathcal{P}} \exp\left(\boldsymbol{w}_{l'}^T \boldsymbol{h}_t + b_{l'}\right)} \left(1 - \frac{1}{\sum_{l' \in \mathcal{P}} \exp\left(\boldsymbol{w}_{l'}^T \boldsymbol{h}_t + b_{l'}\right)}\right) \boldsymbol{w}_l = g_l(\boldsymbol{h}_t) \boldsymbol{w}_l \tag{4.39}$$

where $g_l(\boldsymbol{h}_t)$ is a scalar-valued non-linear function of $\boldsymbol{h}_t$. This result shows that the optimal perturbation direction is in fact $\boldsymbol{w}_l$. However, the scaling of this direction is different to that given in eq. (4.37). The hidden layer activation undergoes a non-linear transformation to yield the scalar-valued factor which multiplies the direction $\boldsymbol{w}_l$. Consequently, the interpretation of α_l setting the level of network stimulation in the transformed space is lost. The effect of the magnitude of the classifier weight $\boldsymbol{w}_l$ is negated in the framework for network perturbation by normalising it in the l_2-norm prior to perturbing in this direction:

$$\Delta \boldsymbol{h}_t = \sum_{l \in \mathcal{P}} \alpha_l \Delta \boldsymbol{h}_{t,l} = \sum_{l \in \mathcal{P}} \alpha_l g_l(\boldsymbol{h}_t) \hat{\boldsymbol{w}}_l. \tag{4.40}$$

Chapter 5

Results and Discussion

5.1 Data

The purpose of this section is to provide a detailed description of the data sets used in both training and evaluation of the GEC systems. The primary aspect in which the data sets differ is in their style, being either written text or transcripts of speech. The transcriptions could be obtained through manual transcriptions (MAN) performed by a human or the output of an automatic speech recognition (ASR) system. A summary of the language corpora is provided in Table 5.1

Cambridge Learner Corpus (CLC) The Cambridge Learner Corpus [27] is an English text corpus collected collaboratively by Cambridge University Press and the University of Cambridge Local Examinations Syndicate. The CLC data consists of written examinations of candidates at different proficiency levels with 86 different first languages (L1s). The examination scripts range across eight different English first language (EFL) examinations and cover both general and business English. Grammatical errors in the erroneous learner text have been manually annotated using a taxonomy of approximately 80 error types. The correct native responses are then transcribed, giving paired data of erroneous and correct sentences. The data set is divided into train and test sets, the former of which is used for GEC training.

The FCE-public data set [37] is a subset of the CLC corpus which has been made publicly available. It consists of 1,244 exam scripts written by candidates taking the Cambridge ESOL First Certificate in English (FCE) examinations between the years 2000 and 2001. The examination assesses the mastery of the upper-intermediate proficiency level (B2 level on the CEFR scale [7]) in learners of English as an additional language through response to

exam prompts eliciting free-text answers. The FCE test set is a held-out subset of the of the corpus, and as such is in-domain (ID) with the training data.

The written text data from the CLC corpus required processing to closer match the form of data from the speech transcriptions. Unlike written data, speech data is free of spelling mistakes, punctuation or capitalisation. Thus, to adapt the written corpora to be closer to speech transcriptions, all spelling mistakes and punctuation were removed from the written data and the text lowercased. Furthermore, sentence start and end symbols were appended to the start and end of the written sentences.

NICT-JCE The NICT Japanese Learner English (JLE) Corpus [16] is a publicly available non-native speech corpus. It comprises of manual transcriptions of an English oral proficiency interview involving Japanese English learners at A1-B2 levels on the CEFR scale. The corpus contains a total of 167 interviews which have been manually annotated with grammatical errors and speech disfluencies. In this project, the NICT data set was used as an out-of-domain (OOD) data set for spoken GEC tasks.

BULATS The Business Language Testing Service (BULATS) [4] is an English language test developed by Cambridge ESOL to assess language skills in a business context. The tests consists of read-aloud and free-speaking sections. In this project, only the latter free-speaking part from 226 candidates was used. The free-speaking part of the test is subdivided into three sections. The first section involves presenting the candidate with a business or work-related topic to talk about for 60 seconds. The second section follows a similar structure; candidates are shown a series of charts and graphs which they are asked to describe for 60 seconds. The third section requires candidates to answer five questions related to a specific scenario and provide short 20 second responses. The data set consists of a total of 1438 responses from these three sections. The first languages of candidates are: Arabic, Dutch, French, Polish, Thai, Vietnamese, and their proficiency level are approximately evenly distributed across CEFR grades A1-C (C1 and C2 are combined). In this project, both manual (BLT-MAN) and ASR (BLT-ASR) transcriptions of BULATS data were used as OOD data sets for spoken GEC. The average ASR word error rate (WER) for this data was 19.5%.

5.2 Experimental Set-Up

Word embeddings for both the encoder and decoder were initialised using 200-dimensional **Global Vectors for Word Representation** (GloVe) word vectors[1] pre-trained on 6 billion

[1] GloVe word vectors: https://github.com/stanfordnlp/GloVe

Table 5.1 Summary of the language corpora.

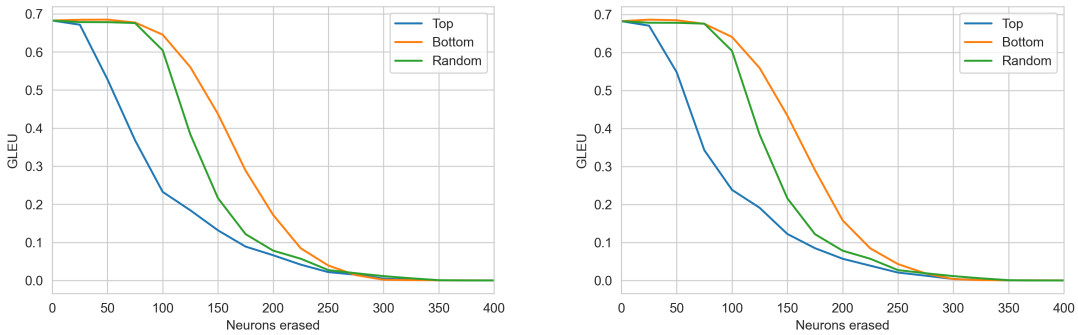
Set	Train	FCE	NICT	BLT-MAN	BLT-ASR
# of sentences	1.8M	2.7K	21.1K	3.7K	3.6K
Avg. Length	13.4	14.0	6.6	16.6	16.7
Style	Written	Written	Spoken	Spoken	Spoken
Domain	Ref.	ID	OOD	OOD	OOD

Table 5.2 GLEU performance of the individual GEC models.

Test Set	**System**						
	1	2	3	4	5	6	Average $\pm$SD
FCE	68.05	67.97	68.25	68.41	68.06	68.13	68.15 ± 0.16
NICT	44.59	44.27	44.80	44.71	44.40	44.46	44.54 ± 0.16
BLT-MAN	47.96	48.04	48.35	48.49	48.13	48.16	48.19 ± 0.20
BLT-ASR	28.99	29.14	29.65	29.36	29.24	28.98	29.23 ± 0.25

word tokens. Two bi-directional RNN layers were used for the encoder and four uni-directional RNN layers for the decoder. All RNN layers were 200-dimensional, matching the dimensionality of the word embeddings. Thus, the concatenated forward and backward hidden layer activations in the encoder $\boldsymbol{h}_t$ were 400-dimensional. Residual connections were applied to improve training efficiency. An attention mechanism with dot-product attention was employed to bridge the encoder and decoder sides of the system. With the exception of the pre-trained word embeddings, all system weights were randomly initialised. The model was trained on the CLC training data using the *cross-entropy loss* with the Adam optimiser [19]. Training was run for 20 epochs with a *batch size* of 256, a *learning rate* of 0.001, a dropout probability of 20% for all hidden layers and batch normalisation.

The network was trained starting from six different random initialisations. For each initialisation, the model configuration at the training epoch with the highest GLEU score on the FCE test set was appended to the ensemble, yielding an ensemble of six models. Table 5.2 outlines the GLEU performance of the individual models in the ensemble for the four test sets of interest. The remainder of the results focus on applying network perturbations to control a single model in the ensemble, System 1 in Table 5.2. Thus, it is this system's results which are taken as the baseline system performance.



(a) Max Corr (b) Min Corr

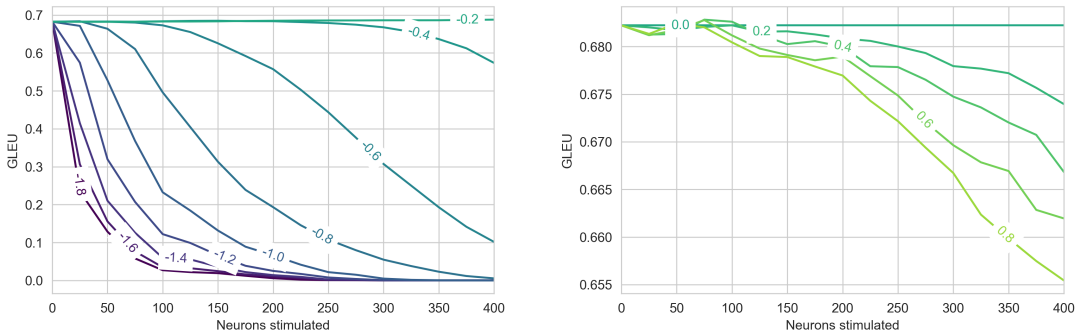
Figure 5.1 Erasing encoder output neurons from the top and bottom of the Max and Min Corr ranked lists. The average effect of erasing neurons from 50 randomly generated lists is also shown.

5.3 Network Perturbation

The hidden layer activations over all words in the CLC training set were recorded for the six GEC models in the ensemble. The Max and Min Corr algorithms were employed to rank the 400 neurons in the encoder output layer by their PCC scores. In order to verify that neurons ranked highly by these correlation algorithms were in fact important for the GEC model, the effect of erasing these neurons was evaluated. The hypothesis was that erasing neurons from the top of the ranked list would impact translation performance to a greater degree than erasing neurons from the bottom, as the higher ranked neurons are of greater importance to the model.

Figures 5.1a and 5.1b plot the GLEU scores of the output translations on the FCE test data set as functions of the number of neurons erased from the top and bottom of the Max and Min Corr ranked lists respectively. The average translation performance obtained by erasing neurons from 50 randomly generated lists is also shown. These results confirm the proposed hypothesis: erasing neurons from the top of the ranked lists impacts translation performance to a greater degree than erasing neurons randomly, which in-turn impacts translation performance to a greater degree than erasing neurons from the bottom of the ranked list. This verifies the importance of the highly-ranked neurons for the task of GEC.

Erasing neurons from the top and bottom of each ranked list impacts GLEU score to a similar extent. This is due to the similarity between the salient neurons identified by each algorithm: 46 of the top 50 neurons in the Max Corr list appear in the top 50 neurons of the Min Corr list. This suggests that neurons which capture features strongly between two models also correlate well with many other models.



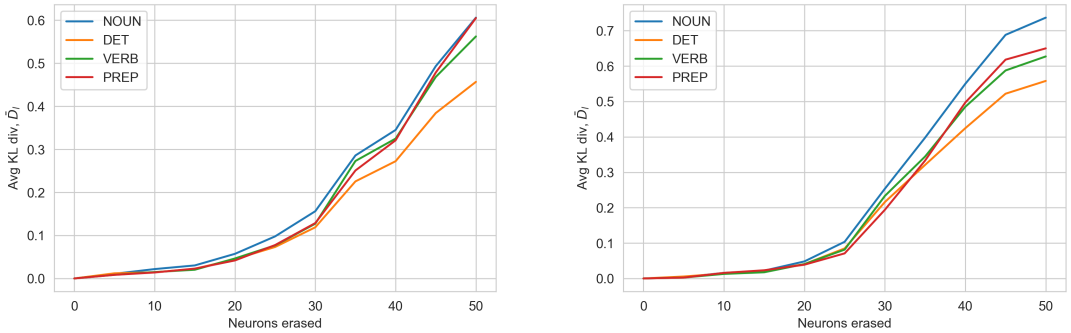
(a) Under-stimulation

(b) Over-stimulation

Figure 5.2 Stimulation of encoder output neurons' from the top of the Max Corr ranked list. Panel (a) plots the GLEU score for network node under-stimulation. Panel (b) plots the GLEU score for network node over-stimulation. The stimulation factor α is shown for each curve. Note the difference in scale on the ordinate axes between the two panels.

The node ablation experiments were extended to investigate the effects of network node stimulation. Figure 5.2 plots the GLEU scores of the output translations on the FCE test data set as functions of the number of neurons stimulated from the top of the Max Corr ranked list for a range of values of α. Due to the large overlap between the rankings, very similar plots were obtained as neurons were incrementally stimulated from the top of the Min Corr ranked list. The results demonstrate that GLEU score is impacted to a far greater extent when neuron activations are under-stimulated compared to when they are over-stimulated. This will be shown to be a recurring theme across many of the results presented in this chapter. There are two possible explanations for this phenomenon. The first is based on the model configuration. The model was trained using batch normalisation. However, at inference time there is no dynamic implementation of batch normalisation. As such, variations in node stimulation may be asymmetrical about $\alpha = 0$. The second, more likely reason, is attributed to the nature of the hyperbolic tangent activation function used in the RNN LSTM units. The function saturates when its argument is very positive or very negative, meaning the function becomes very flat and insensitive to small changes in its input. Over-stimulating pushes the hidden layer activations into the saturated region. This results in smaller network perturbations being propagated to the decoder, and so smaller degradations to the output translation. Conversely, the function is pushed into the linear region for under-stimulation, resulting in larger network perturbations being propagated to the decoder, and thus larger degradations to the output translation.

The previous investigations focused on unsupervised methods for network node perturbations: hidden layer activations were correlated over the entire data set to give neurons important for the GEC model's overall operation. The natural progression is asking whether



(a) NOUN (b) DET

Figure 5.3 Average PoS KL divergence with neuron erasure from the top of: (a) the noun Max Corr ranked list, and (b) the determiner Max Corr ranked list.

or not it is possible to find individual nodes which are responsible for the grammatical error correction of particular parts of speech.

The input words $\boldsymbol{w}_{1:T}$ in the CLC training set were annotated with PoS labels $\boldsymbol{l}_{1:T}$ using the Stanford Neural Network Dependency Parser, as described in Chapter 2.3.2. These labels were used as proxy labels for the hidden layer activation vectors $\boldsymbol{h}_{1:T}$. The hidden layer activations were then grouped by PoS. The Max Corr algorithm was run on the hidden layer activations for a single PoS group at a time, yielding a ranked list of encoder output neurons for that PoS. The impact of neuron erasure was evaluated according to eq. (3.17): the KL divergence was computed between the predicted probability mass functions of the unperturbed and perturbed models, and averaged over each target PoS to yield $\bar{D}_l$.

Figures 5.3a and 5.3b plot the average KL divergence values for four different parts of speech (noun, determiner, verb and preposition) as functions of the number of neurons erased from the top of the noun and determiner Max Corr ranked lists. These plots reveal that erasing neurons identified as being important for one PoS impacts the GEC model over many parts of speech. The assumption that individual nodes can be turned on and off to give PoS control does not hold for the task of GEC. This comes as an effect of the highly distributed nature of information within the DNN: the system is trained for generalisation through use of techniques such as dropout, resulting in individual neurons being responsible for the encoding of multiple parts of speech.

As network node perturbation did not yield the levels of model control required for this project, the focus was shifted towards network direction perturbations. The most logical starting point for network direction perturbation is finding the direction $\Delta\boldsymbol{h}_t$ which maximises the change in the network's prediction for the target word. Equipped with supervised training or test data, consisting of an input sequence of T words $\boldsymbol{w}_{1:T}$ and an K-length target sequence

$\boldsymbol{y}_{1:K}$, this direction can be formalised mathematically as the gradient of the prediction of the ground truth symbol y_i with respect to the hidden layer activation. The perturbation direction corresponding to the i-th output word is then:

$$\Delta \boldsymbol{h}_t = \frac{\alpha}{Z} \left. \frac{\partial P\left(\hat{y}_i = y_i \mid \boldsymbol{h}_{1:T}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}\right)}{\partial \boldsymbol{h}_t} \right|_{\boldsymbol{h}_{1:T}}, \tag{5.1}$$

where the normalising constant Z enforces the vector of derivatives to be unit magnitude in the l_2-norm:

$$Z = \left\| \left. \frac{\partial P\left(\hat{y}_i = y_i \mid \boldsymbol{h}_{1:T}, \boldsymbol{y}_{<i}; \boldsymbol{\theta}\right)}{\partial \boldsymbol{h}_t} \right|_{\boldsymbol{h}_{1:T}} \right\|_2. \tag{5.2}$$

Normalising the derivative vector to be unit magnitude enables the degree of perturbation to be set entirely by the scalar-valued factor α under the framework for network perturbation.

The change in the network's predictions resulting from perturbation in the derivative directions was quantified by computing the KL divergence between the baseline and perturbed models, and averaging over all output predictions in the data set. The hypothesis was that perturbing the model in the derivative directions in eq. (5.1) would result in more significant changes to the network's predictions than purely random directions, thus giving larger values of average KL divergence relative to random directions.

Figure 5.4 plots the average KL divergence as a function of the level of perturbation applied α in the gradient directions for the FCE test set. Also shown is the average KL divergence obtained by perturbing the model in random directions. These results demonstrate that perturbing the hidden layer activations in the gradient directions is an effective framework for modifying the network's predictions: the average KL divergence is increased to a much larger degree when perturbed in the gradient directions than purely random directions. Thus, the derivative directions are deemed important, in the sense that perturbing in these directions results in substantial shifts to the model's predictions.

Whilst it is interesting to consider how the network parameters change by perturbing in isolated word directions, the gradient-based method in this form is not feasible for control at inference time: the reference words for the outputs were given as part of the supervised data set. This is not the case when making predictions on unseen data, where the true class labels are unknown. Furthermore, the perturbation directions obtained are specific to the particular instance of input words and target sequence. The recurrent nature of the sequence-to-sequence RNN model means that the sequence of hidden layer activations $\boldsymbol{h}_{1:T}$ and target network outputs $\boldsymbol{y}_{1:K}$ are unique to the data set used. As a result, the derivative direction is too unique to the data set. Thus, the perturbation direction computed for the i-th

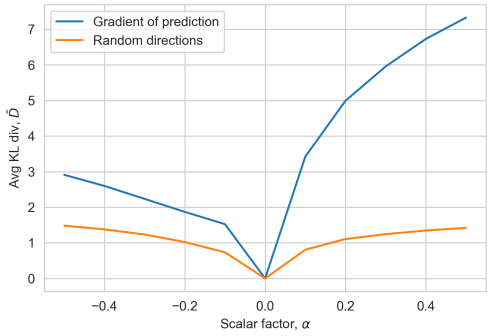


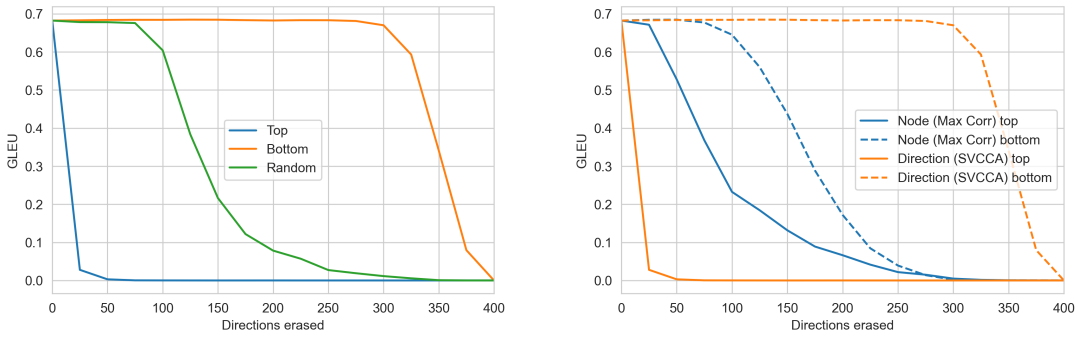
Figure 5.4 Average KL divergence with scaling of the gradient and random perturbation directions.

target word y_i in one data set is not the optimal perturbation direction for that same word under a different data set.

In pursuit of universal network perturbation directions, the SVCCA algorithm was employed to correlate encoder output directions for the hidden layer activations over all words in the CLC training set over the six GEC models in the ensemble, returning a ranked list of network directions. In an entirely equivalent way to node ablation, the importance of these directions was evaluated by incrementally erasing directions from the top and the bottom of the SVCCA ranked list. The hypothesis was that erasing directions from the top of the ranked list would impact translation performance to a greater degree than erasing directions from the bottom, as the higher ranked directions are of greater importance to the model.

Figure 5.5a plots the GLEU score of the output translation for the FCE test data set as a function of the number of directions erased from the top and bottom of the SVCCA ranked list. The average translation performance obtained by erasing directions from 50 randomly generated orthonormal projection matrices is also shown. These results verify the proposed hypothesis: erasing directions from the top of the SVCCA ranked list degrades translation performance to a far larger degree than erasing directions randomly, which in-turn degrades translation performance to a greater extent than erasing directions from the bottom of the ranked list.

One quite remarkable finding from this experiment is the number of SVCCA directions required for high GEC performance. The GLEU curve remains flat with the erasure of the bottom 300 SVCCA directions, and falls rapidly as the top 100 directions are erased. The bottom 300 directions (out of a total of 400) can be erased at the expense of only a 0.01 decrease in GLEU score compared to the baseline model, a percentage decrease of just 1.80%. This suggests that GEC models do not rely on the full dimensionality of the encoder layer in



(a) (b)

Figure 5.5 Panel (a) plots the GLEU score with incremental erasure of network perturbation directions from the top and bottom of the SVCCA ranked list. The average effect of erasing directions from 50 randomly generated lists is also shown. Panel (b) plots the GLEU score with incremental erasure of network perturbation directions from the top and bottom of the Max Corr and SVCCA ranked lists.

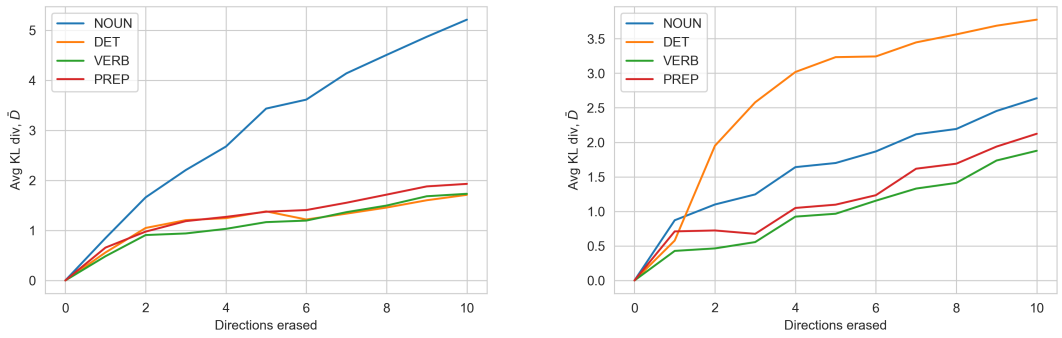
order to perform the task well: the useful information captured by all 400 encoder output neurons is well summarised by the subspace formed by the top 100 SVCCA directions alone.

Figure 5.5b compares the effect of erasing the top/bottom network SVCCA directions to the erasure of the top/bottom Max Corr nodes. It highlights the fact that SVCCA subspace directions are disproportionately important to the representation learned by the encoder output layer relative to the neuron-aligned Max Corr directions: translation performance deteriorates at a much faster rate with erasure of the top network directions than the top network nodes.

To obtain network perturbation directions for a different parts of speech, SVCCA was run on the sub-set of hidden layer activation vectors for each proxy label. The saliency of the returned PoS directions was evaluated by incrementally erasing the number of directions, starting from the top of the ranked list. The impact of direction erasure was assessed through computation of the average KL divergence $\bar{D}_l$ over each target PoS.

Figures 5.6a and 5.6b plot the average KL divergence values for four different parts of speech as functions of the number of directions erased from the top of the SVCCA noun and determiner ranked lists respectively. These results demonstrate that erasing the top 10 SVCCA directions for a given PoS is an effective way of increasing the KL divergence for that PoS and not others, suggesting that salient directions have in fact been returned by the SVCCA procedure.

Whilst erasing directions is a useful tool for verifying the importance of particular network perturbation directions, complete erasure of directions is particularly impactful on the performance of GEC: erasing the top 10 SVCCA noun directions degrades the GLEU



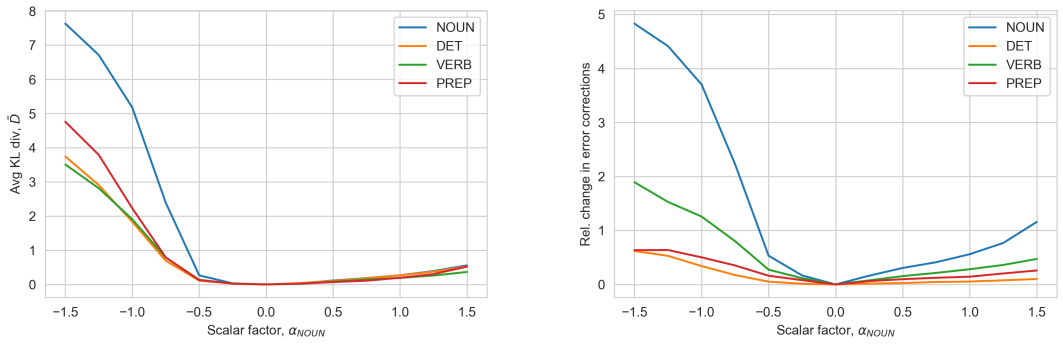
(a) NOUN (b) DET

Figure 5.6 Average PoS KL divergence with erasure of directions from the top of: (a) the SVCCA noun ranked list, and (b) the SVCCA determiner ranked list.

score of the output translation from 68.22 to 31.17. This is due to fact that direction erasure is equivalent to applying a very large network perturbation ($\alpha = -1$). Thus, the probability mass function is quite significantly perturbed, drastically affecting the model's prediction for the error corrected sequence. Having demonstrated that the erasure of the top 10 SVCCA directions for a given PoS is sufficient to increase the KL divergence of that PoS and not others, the level of perturbation applied in these directions α was varied. The purpose of this investigation was to better understand the effects of network direction perturbation on the model, with the aim of increasing the KL divergence for a given PoS whilst impacting the overall translation to a lesser degree. It was hoped that in doing so, the model could be perturbed whilst still obtaining a reasonable GEC output.

Figure 5.7a plots the KL divergence between the unperturbed and perturbed models averaged over four parts of speech as a function of the level of perturbation applied in the top 10 SVCCA noun directions α_{NOUN}. The perturbations in the SVCCA directions for all other parts of speech were set to zero. Under-stimulating the top 10 noun SVCCA directions ($\alpha_{\text{NOUN}} < 0$) appears to be an effective way of controlling the the extent to which the probability mass functions for each PoS are perturbed. Larger perturbations yield larger values of the KL divergence averaged over all target nouns, indicating that the probability mass functions corresponding to noun predictions are perturbed to a greater extent. There are much smaller increases in the KL divergence values averaged over each of the three other parts of speech, implying that the predictions for these parts of speech are perturbed to a far lesser extent than the noun predictions.

The KL divergence metric is instrumental for quantifying the extent to which the network's predictions are modified. However, as a measure of the distance between two distributions, it provides no information on whether the desired part of the probability mass



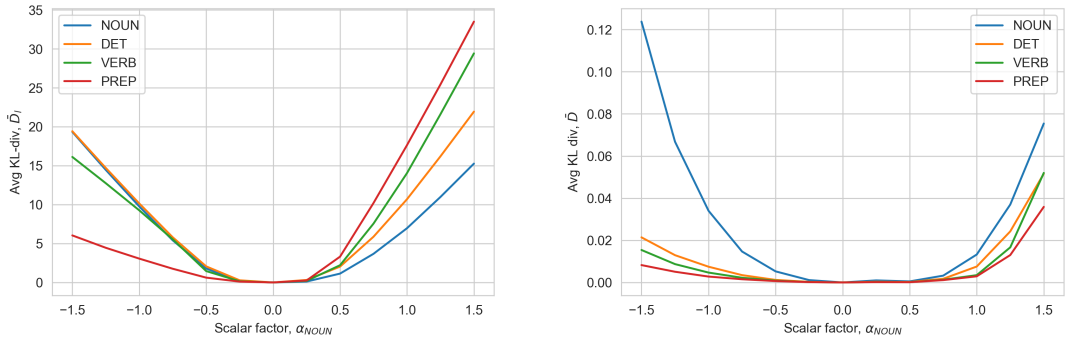
(a) (b)

Figure 5.7 Panel (a) plots the average PoS KL divergence with stimulation of the top 10 SVCCA noun directions. Panel (b) plots the relative change in errors between the baseline and perturbed models for the stimulation of the same 10 perturbation directions.

function is shifted, or whether the shift is for words outside the domain of interest. For instance, when perturbing in the SVCCA noun directions, it is unclear whether it is the predictions for nouns which are altered, or words belonging to different parts of speech. To better ascertain how the network's predictions change under network direction perturbation, the relative number of errors between the unperturbed and perturbed models was calculated for each PoS. Figure 5.7b plots the relative number of errors for four parts of speech as a function of the level of perturbation applied in the top 10 SVCCA noun directions α_{NOUN}. Under-stimulating the SVCCA noun directions drastically increases the number of noun error corrections generated by the perturbed GEC model, whilst keeping the number of additional grammatical error corrections for other parts of speech low. This suggests that it is in fact the predictions for the nouns which change by perturbing the model in the SVCCA noun perturbation directions.

The remainder of the results presented in this section are associated with the use of the logistic classifier for obtaining network perturbation directions. The primary advantage of using the logistic classifier over correlation-based approaches is procurement of a posterior probability mass function built in the activation function space. With a clear interpretation of the classifier weights as the vector orthogonal to the decision boundaries which separate decision regions, the precise form of the network direction perturbation can be investigated.

The first form of perturbation trialled was that of the fixed bias, as given by eq. (4.36). Figure 5.8a plots the KL divergence between the unperturbed and perturbed models averaged over four parts of speech as a function of the level of the perturbation bias applied in the top 10 logistic classifier noun directions α_{NOUN}. The KL divergence is increased to a similar extent over all four parts of speech, indicating that the fixed bias approach does not yield a

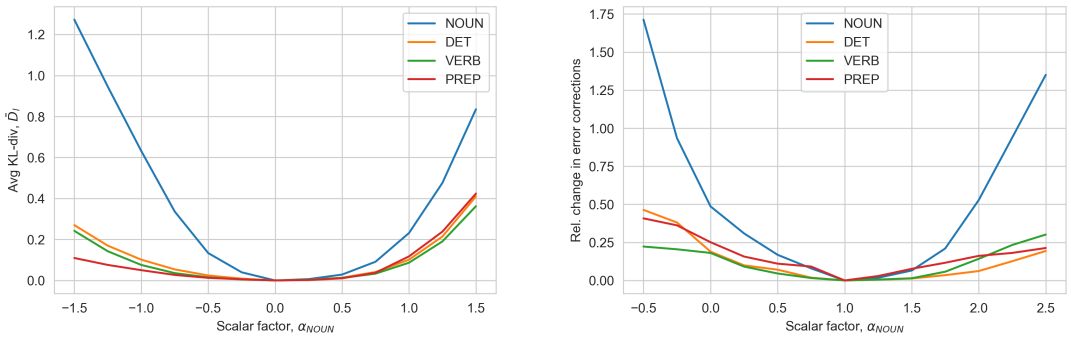


(a) (b)

Figure 5.8 Average PoS KL divergence with stimulation of the top 10 logistic classifier noun directions via: (a) fixed bias perturbation, and (b) projection onto the classifier directions.

controllable system. The logical progression is to make the network direction perturbation a function of the hidden layer activation. Thus, the directions $\hat{\boldsymbol{w}}_l$ in which the hidden layer activations were perturbed were scaled by an amount given by the projection of $\boldsymbol{h}_t$ onto $\hat{\boldsymbol{w}}_l$, as in eq. (4.37). Figure 5.8b plots the average PoS KL divergence's obtained by perturbing the model in the same 10 logistic classifier noun directions under this new scheme. Under-stimulating the noun directions greatly increases the KL divergence averaged over all nouns between the unperturbed and perturbed GEC models, whilst keeping the KL divergence averaged over all other parts of speech low. This is exactly the behaviour required of a controllable model, thus demonstrating that salient directions have in fact been returned by the logistic classifier, and that a functional framework for control has been established.

The final form of perturbation trialled was that of the softmax derivative perturbation given by eq. (4.40). Figure 5.9a plots the average PoS KL divergence obtained by perturbing the model in the 10 logistic classifier noun directions under this construction. Once again, under-stimulation is shown to yield a highly controllable GEC model. In order to confirm that the correct portion of the probability mass function was being shifted, the relative number of error corrections between the perturbed and unperturbed models was computed for each PoS. Figure 5.9b plots the relative number of error corrections for four parts of speech as a function of the level of perturbation applied in the same 10 logistic classifier noun directions as Figure 5.9a. It shows similar results as those for SVCCA: noun predictions are changed by perturbing the model in the logistic classifier noun directions, thus verifying that the correct portion of the probability mass function is shifted by the perturbations.



(a) (b)

Figure 5.9 Panel (a) plots the average PoS KL divergence with stimulation of the top 10 logistic classifier noun directions via the softmax derivative. Panel (b) plots the relative change in errors between the baseline and perturbed models for the stimulation of the same 10 perturbation direction.

5.4 Domain Adaptation

The previously network perturbation experiments were applicable in asserting whether or not given perturbation directions could be used for model control. However, the different interpretations of the scalar-valued factors α_l for different control schemes meant that the results could not be compared across methods: the same value of α_l produced perturbations of different overall magnitudes for different perturbation methods. A means of comparing different perturbation schemes is required to assess their performance against one another and the baseline (unperturbed) model. A task-based approach was adopted in order to achieve this.

Having trained the GEC model in the written text domain on the CLC training data, a form of domain adaptation was performed, in which the model was fine-tuned to the target speech domain. In this case, fine-tuning consisted of the optimisation process which involves finding the set of scalar factors $\boldsymbol{\alpha} = \{\alpha_l\}_{l \in \mathcal{P}}$ which maximised the probability of observing the target data under the perturbed model, holding the original model parameters fixed:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \left\{ P\left(y_{1:K} \mid w_{1:T}; \boldsymbol{\alpha}, \boldsymbol{\theta}\right) \right\}. \tag{5.3}$$

The intuition behind this learning process was that different data sets have different statistics of PoS errors, and hence when shifting domain, the importance of each PoS may change from the written text domain on which the system was trained. The learned scalar factors will reflect this, being the optimal degrees of perturbation in the hidden layer activation space for each data set.

Different control schemes were then assessed by evaluating the GLEU scores of the fine-tuned models for the various data sets. Control schemes that produce PoS directions of greater saliency should be better able to adapt to different data sets, particularly those which are OOD with the training data. Consequently, they should have higher translation accuracies, and so higher GLEU scores. Comparing the GLEU score of the fine-tuned models to that of the baseline model gives a direct measure of the ability of the PoS directions, and hence perturbation scheme, to adapt to the data set.

The optimal scalar factors were found for the top 29 parts of speech by frequency in the training set for the four different test sets. Covering a broad range of parts of speech enables for adaptation throughout the distribution of data. With a small amount of annotated speech data in each test set, an ensemble of models was created for each fold in a 10-fold cross validation. Performance was evaluated against the test set by averaging the GLEU scores over the 10 models. Table 5.3 reports the fine-tuning results for the four different test sets. GLEU scores are shown for fine-tuning of the top 10 Max Corr, SVCCA and logistic classifier directions. The classifier directions were fine-tuned using two control schemes: the projection of the hidden layer activations onto the classifier weights (LClass 1) and the softmax derivative construction (LClass 2). On average, there was a 10.2% decrease in GLEU score following the optimisation of MaxCorr neuron activations. This further validates that network node stimulation is an ineffective way of controlling the GEC model. One the other hand, there was a 1.20% improvement in GLEU score following the optimisation of SVCCA directions, a 0.396% increase for the optimisation of LClass 1 (projection) perturbations, and a 0.327% increase for the optimisation of LClass 2 (derivative) perturbations, with the greatest changes observed when the model was adapted to OOD data sets. Although these results demonstrate a degree of domain adaptation, the improvements are somewhat low. To better understand what issues were potentially at play, two questions were raised: (1) Has the system done what it has been asked to do?; and (2) Has the system done what one wants the system to do?

The remainder of the investigations sought to answer both of these questions through various explorations into the domain adaptation of the logistic classifier directions, perturbed using the first control scheme (LClass 1). The classifier directions can be refined in a number of ways through modification of the classifier training set-up, enabling the first question to be probed in detail. Perturbing the system using the first control scheme shares symmetries with node stimulation and the style of perturbation used in SVCCA. Hence, use of this scheme allows for a more general analysis than if directions are perturbed using the second (gradient) control scheme.

Table 5.3 GLEU scores for the baseline and fine-tuned GEC models.

Test Set	**System**				
	Baseline	Max Corr	SVCCA	LClass 1	LClass 2
FCE	68.05	61.89	68.23	68.21	68.20
NICT	44.59	45.58	46.34	44.87	44.96
BLT	47.96	37.38	48.01	47.94	48.00
BLT-ASR	28.99	25.46	29.22	29.32	29.05

One potential reason why the gains for optimising the models in Table 5.3 are not more significant is due to overlap in perturbation directions for different parts of speech. Consider two distinct parts of speech with labels l and l', with corresponding perturbation directions $\Delta\boldsymbol{h}_{t,l}$ and $\Delta\boldsymbol{h}_{t,l'}$. Should there be shared representations between $\Delta\boldsymbol{h}_{t,l}$ and $\Delta\boldsymbol{h}_{t,l'}$, then the interaction between perturbation directions will hinder the extent to which the level of perturbation for each PoS can be optimised. A solution to this is to enforce an orthogonality constraint in the offsets. For the logistic classifier, this amounts to adding a regularisation term which penalises overlaps in the directions of classifier weights:

$$\hat{\boldsymbol{\theta}}^c = \underset{\boldsymbol{\theta}^c}{\operatorname{argmax}} \left\{ \sum_{t=1}^{T} \log\left(P(l_t|\boldsymbol{h}_t;\boldsymbol{\theta}^c)\right) - \lambda \sum_{l\in\mathcal{P}} \sum_{l'\neq l} |\boldsymbol{w}_l \cdot \boldsymbol{w}_{l'}| \right\}, \tag{5.4}$$

where the scalar valued hyper-parameter $\lambda > 0$ determines the relative importance of the log-likelihood term and the orthogonality constraint term: larger values of λ place greater significance on the orthogonality constraint term and less significance on the log-likelihood term. An appropriate value of λ was selected as a trade-off between maximising the log-likelihood and minimising the direction overlap, one which in general results in a similar dynamic range for each term.

Having learned the regularised classifier directions under the augmented objective function in eq. (5.4), the network was perturbed in these directions and the optimal levels of perturbation found for the four different test sets. Table 5.4 shows the GLEU scores for the baseline GEC model, the model fine-tuned on the MLE logistic classifier directions (LClass), and the system fine-tuned on the logistic classifier directions learned with the orthogonality constraints (LCLass + Orth.). There was a 0.396% increase for system fine-tuned on the MLE of classifier parameters averaged over the four test sets, compared to a 0.390% average increase for the regularised learned classifier parameters. These results demonstrate that the unconstrained classifier directions are of greater saliency than the constrained directions: reducing the overlap between classifier directions hindered the ability of the fine-tuned model

Table 5.4 GLEU scores for the baseline and fine-tuned logistic classifier GEC models.

Test Set	**System**		
	Baseline	LClass	LClass + Orth.
FCE	68.05	68.21	68.16
NICT	44.59	44.87	44.87
BLT	47.96	47.94	48.00
BLT-ASR	28.99	29.32	29.30

Table 5.5 Two sample original and corrected sentence pairs, annotated with grammatical error type information extracted and classified by ERRANT. The verb reversed target set is constructed by undoing the verb correction made from `sit` $\rightarrow$ `sat`.

Input, $\boldsymbol{w}_{1:12}$	the	cat	sit_1	on	the	mat
	they	recommend	running	at_2	the	morning
Target, $\boldsymbol{y}_{1:12}$	the	cat	sat_1	on	the	mat
	they	recommend	running	in_2	the	morning
$ERRANT(\boldsymbol{w}_{1:12}, \boldsymbol{y}_{1:12})$	-	-	verb_1	-	-	-
	-	-	-	prep_2	-	-
Reversed target, $\boldsymbol{y}^*_{1:12}$	the	cat	sit_1	on	the	mat
	they	recommend	running	in_2	the	morning

to adapt itself to OOD data. Thus, the optimal directions for network perturbation are those which maximise the log-likelihood of the training data in the hidden layer activation space.

Having shown the MLE classifier directions to be effective for model control in the virtual task of network perturbation, and the addition of orthogonality constraints to not improve fine-tuning performance on the test sets, it was concluded that the perturbation directions were not responsible for the poor domain adaptation performance. Consequently, attention was turned to answering the first of the questions raised: has the system done what it has been asked to do? To better answer this question, a series of *reversed* data sets were developed. For a the supervised FCE test data set $\mathcal{D} = \{\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:K}\}$, ERRANT was run between the input and target sequences, revealing the position and nature of the target grammatical corrections. A PoS with label $l \in \mathcal{P}$ was selected, and all corrections associated with l undone from the target sequence. Table 5.5 details an example for the construction of a modified sequence of targets $\boldsymbol{y}^*_{1:L}$ for a verb reversed data set given two sample parallel input-output sentences.

The MLE of parameters $\boldsymbol{\alpha} = \{\alpha_l\}_{l \in \mathcal{P}}$ was then found for the reversed data set $\mathcal{D}^* = \{\boldsymbol{w}_{1:T}, \boldsymbol{y}^*_{1:L}\}$. Working on the assumption that a controllable GEC model can be made to correct grammatical errors for a given PoS and not others, the aim was to learn a set of scalar factors which would not induce grammatical corrections for the chosen PoS, but continue to correct grammatical errors for all other parts of speech. The use of reversed data sets

Table 5.6 Percentage change in grammatical error type corrections following fine-tuning on PoS reversed test sets.

Test set	NOUN	DET	VERB	PREP
FCE	8.54	-1.26	9.01	0.00
NOUN rev.	**8.54**	-1.01	9.01	0.00
DET rev.	8.54	**-1.01**	8.12	0.38
VERB rev.	9.55	-1.01	**8.12**	0.38
PREP rev.	9.55	-1.01	8.12	**0.78**

truly challenges the framework proposed for network perturbation; if the system fails on the artificial data sets, then it is unlikely to perform well on real data sets.

The metric for assessing the ability of the model to adapt to a reversed data set was the relative change in the number of grammatical error corrections between the baseline and fine-tuned models. Let $\hat{N}_l$ and $\tilde{N}_l$ denote the number of grammatical errors corrected by the baseline and fine-tuned models for the PoS with label l. A fine-tuned system which achieves perfect performance would return a relative change of -1 corrections for the PoS reversed in the synthetic data set, correcting 0 grammatical errors of this kind ($\tilde{N}_l = 0$), and a relative change of 0 for all other parts of speech, correcting an equal number as the baseline model ($\tilde{N}_{l'} = \hat{N}_{l'}$).

Table 5.6 reports the results for 10-fold cross-validation fine-tuning on the reversed data sets for four different parts of speech. To better match the scale of numbers involved, the relative change in corrections are expressed as percentage changes. The percentage change in error type corrections following fine-tuning of the FCE test set is shown for reference. The system demonstrated an inability to adapt to the artificial data sets. The percentage change in PoS corrections on the reversed data sets were almost identical to the percentage change for the original FCE test set. Though these results are disappointing from a performance perspective, they are incredibly useful from a research point of view. The conclusion that the classifier directions were suitable perturbation directions for network control ruled out the possibility that a flaw lay in the directions themselves. The artificial data set results reveal an issue in the way the system is perturbed, as the system is not doing what it has been asked to do.

In its current form, the perturbations applied to the encoder output layer are linear functions of the hidden layer activations: $\boldsymbol{h}_t$ is linearly transformed by a projection matrix, elements scaled in the transformed space, and the scaled vector linearly transformed back into the original activation space, as shown in Figure 4.2. Information is propagated from the encoder to the decoder side through the attention mechanism. This involves computing

a context vector $\boldsymbol{c}_i$, a weighted average of the hidden states $\boldsymbol{h}_t$. The weightings $\alpha_{i,t}$ obey a sum-to-one constraint, as given by eq. (2.9).

The combination of a linear perturbation and a linear averaging process with a sum-to-one constraint begs the question of how the encoder perturbations alter the operation of the attention mechanism, and by extension the decoder. As the weights $\alpha_{i,t}$ sum-to-one, the averaging process performed by the attention mechanism could effectively result a single, average perturbation being applied to all hidden layer activations on the encoder side. This would result in a fixed offset being applied to the first layer of the decoder. The results presented in Figure 5.8a demonstrated that a fixed offset perturbation on the encoder side was ineffective for model control. Instead, the perturbation had to be made a function of $\boldsymbol{h}_t$. This suggests that a global shift to the decoder hidden states is also sub-optimal for network control. In a similar way to the encoder, the perturbation $\Delta\boldsymbol{h}_t$ could be made a function of the decoder hidden state $\boldsymbol{s}_i$, such that when the perturbation is applied, there is an increase in the probability of generating a PoS correction.

The major drawback to this approach is the substantial increase in the complexity of the analysis. Dealing with two different hidden states spanning both sides of the model is far more complex than just one hidden state on the encoder side. It also risks losing the valuable connection between the network perturbation $\Delta\boldsymbol{h}_t$ and the t-th input word w_t.

More work is required to investigate the effects of the encoder perturbation on the attention mechanism and decoder side. The predicted probability mass function at the decoder is the primary element of interest in the analysis; it does not matter how the model is controlled, so long as the network predictions can be changed. Hence, investigating the decoder is the natural progression in the timeline of work.

Chapter 6

Conclusions and Future Work

This project sought to address the challenging task of improving network interpretability for deep learning GEC systems. The analysis focused on the encoder output layer of an RNN sequence-to-sequence architecture. In particular, the line of work was geared towards uncovering network perturbations that could be applied to control the outputs of the system in a predictable and reliable way. The motivation behind this was that if equipped with an understanding of how the encoder output layer was operating, the information could be used to manipulate the system's predictions for particular parts of speech.

The framework for network perturbation remained fixed throughout the project. What varied was the way in which the perturbation directions were constructed. The first approach employed an ensemble correlation analysis to find individual nodes of high saliency within a given network. The impact of erasing these nodes was assessed on the system. It was shown that erasing more highly-ranked neurons degraded GLEU performance on the FCE test set to a far greater extent than lower-ranked neurons, thus confirming their importance for the task of GEC. An attempt was then made at controlling the system for particular parts of speech through the erasure of carefully selected network nodes. This proved to be very limited for the GEC system: it was not possible to control the network's predictions for a given PoS and not others. This was attributed to the highly distributed nature of information in DNNs, which meant that the individual nodes did not operate in isolation.

Network node perturbation was generalised to network direction perturbation. The aim was to establish individual directions that strongly influenced the performance of the system. The gradient-based analysis showed that perturbing the model in the direction which maximised the change in the network's prediction of the ground truth label was an effective way of altering the predicted probability mass function at the decoder. This motivated the subsequent investigations, which sought to find directions that strongly influenced GEC for particular parts of speech.

In a return to ensemble correlation analysis, SVCCA was used to find linear combinations of neurons that were strongly correlated across models. The impact of erasing these directions was assessed on the GEC system. This revealed that the subspace directions found by SVCCA were disproportionately important to the representation learned by the encoder output layer relative to the neuron-aligned directions.

The SVCCA PoS directions and logistic classifier weights were effective perturbation directions for PoS control. Erasing directions for a given PoS impacted the system's predictions for that PoS with little change to others. The degree of network perturbation was varied by changing the level of stimulation applied to a fixed number of PoS directions. The results showed that the system's predictions for that PoS were altered accordingly. The predicted sequences from the baseline and perturbed models were compared to the input sequence. Through error type annotation, evaluation, and accumulation of statistics, it was shown that the network perturbation in the PoS directions increased the number of grammatical error corrections for a given PoS to a far greater extent than for all others. This verified that the perturbation directions were suitable for system control.

The task of domain adaptation was adopted to assess different perturbation directions and schemes against one another. The optimal level of perturbation was learned for each PoS on OOD speech data. The fine-tuned models were then assessed on the test data. The GLEU score improvements were disappointingly low: on average, there was a 1.20% improvement in GLEU score following the optimisation of SVCCA directions, and a 0.396% increase for the optimisation of logistic classifier directions. The fact that the improvements were small was unlikely to be due to the directions themselves, as: (1) the directions were shown to be important in the previous network perturbation experiments; and (2) adding orthogonality constraints to the logistic classifier directions did not improve fine-tuning performance on the test sets. More likely, the issue was with how the perturbation on the encoder side was being propagated to the decoder.

The natural choice for a future work direction is the investigation of the perturbation propagation issue. A sensible starting point would be looking at how the application of network perturbations impacts the behaviour of the attention mechanism, as it is the attention mechanism that dictates how information is propagated from the encoder to the decoder. In its current form, the perturbation is a function only of the encoder hidden state and perturbation directions. However, there is no reason why the perturbation cannot be a function of any number of network hidden states. It could be made a function of the context vector, incorporating information of how the attention mechanism is operating, and also a decoder hidden state, incorporating information of how the decoder is operating. To do

so will require an interpretability analysis of the attention module and decoder side of the system, thus extending the interpretability analysis performed on the encoder in this report.

An entirely different approach involves taking a step back and questioning whether parts of speech were the right features to have focused on. The network was not trying to achieve corrections through parts of speech- it was finding grammatical errors in the learner text and correcting for those. Thus, the PoS labels for the hidden layer activations could be swapped for error correction labels, and perturbation directions found that are responsible for particular types of error correction. This is a straightforward extension of the work presented in this report, but might prove effective in improving performance on the task of GEC.

References

[1] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, page arXiv:1409.0473.

[2] Bau, A., Belinkov, Y., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. R. (2018). Identifying and Controlling Important Neurons in Neural Machine Translation. *CoRR*, abs/1811.01157.

[3] Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

[4] Chambers, L. and Ingham, K. (2011). The BULATS online speaking test. *Research Notes*, page 21–25. [Online]. Available: https://www.cambridgeenglish.org/Images/23161-research-notes-43.pdf.

[5] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR*, abs/1409.1259.

[6] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

[7] Council of Europe (2001). *Common European Framework of Reference for Languages: Learning, teaching, assessment*. Cambridge University Press.

[8] Dalvi, F., Durrani, N., Sajjad, H., Belinkov, Y., Bau, A., and Glass, J. R. (2018). What Is One Grain of Sand in the Desert? Analyzing Individual Neurons in Deep NLP Models. *CoRR*, abs/1812.09355.

[9] Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Found. Trends Signal Process.*, 7(3–4):197–387.

[10] Durrani, N., Sajjad, H., Dalvi, F., and Belinkov, Y. (2020). Analyzing Individual Neurons in Pre-trained Language Models. *CoRR*, abs/2010.02695.

[11] Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical Correlation Analysis: An Overview with Application to Learning Methods. *Neural Computation*, 16(12):2639–2664.

[12] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385.

[13] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[14] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[15] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167.

[16] Izumi, E., Uchimoto, K., and Isahara, H. (2004). The NICT JLE Corpus: Expliting the language learners' speech database for research and education. *International Journal of the Computer, the Internet and Management*, 12(2):119–125.

[17] Kalchbrenner, N. and Blunsom, P. (2013). Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

[18] Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and Sayres, R. (2017). Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). *arXiv e-prints*, page arXiv:1711.11279.

[19] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980.

[20] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical Phrase-Based Translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

[21] Li, J., Monroe, W., and Jurafsky, D. (2016a). Understanding Neural Networks through Representation Erasure. *CoRR*, abs/1612.08220.

[22] Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. E. (2016b). Convergent Learning: Do different neural networks learn the same representations? In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[23] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, number 9, pages 1045–1048. International Speech Communication Association.

[24] Miller, T. (2017). Explanation in Artificial Intelligence: Insights from the Social Sciences. *arXiv e-prints*, page arXiv:1706.07269.

[25] Morcos, A. S., Barrett, D. G. T., Rabinowitz, N. C., and Botvinick, M. (2018). On the importance of single directions for generalization. *arXiv e-prints*, page arXiv:1803.06959.

[26] Napoles, C., Sakaguchi, K., Post, M., and Tetreault, J. (2015). Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

[27] Nicholls, D. (2003). The Cambridge Learner Corpus - error coding and analysis for lexicography and ELT. *Proc. of the Corpus Linguistics 2003 conference; UCREL technical paper number 16*.

[28] 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 of the Association for Computational Linguistics*, pages 311–318.

[29] Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6076–6085.

[30] Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[31] Shriberg, E. E. (1994). *Preliminaries to a Theory of Speech Disfluencies*. PhD thesis, University of California at Berkeley.

[32] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[33] Swietojanski, P., Li, J., and Renals, S. (2016). Learning Hidden Unit Contributions for Unsupervised Acoustic Model Adaptation. *CoRR*, abs/1601.02828.

[34] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[35] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

[36] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv e-prints*, page arXiv:1609.08144.

[37] Yannakoudakis, H., Briscoe, T., and 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*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

[38] Zhou, B., Sun, Y., Bau, D., and Torralba, A. (2018). Revisiting the Importance of Individual Units in CNNs via Ablation. *CoRR*, abs/1806.02891.

Appendix A

Risk Assessment

The project was entirely computer-based, requiring large amounts of screen use and desk work. Hence, the risks presented were from ergonomic issues and eye strain. The risk assessment evaluated at the beginning of this project reflected these issues well and the hazards were appropriately handled. The first risk was reduced by obeying standard seating posture guidelines. The second risk was minimised through a strict policy of regular breaks. Furthermore, the workstation was set up to conform with the Display Screen Equipment regulations. There were no further risks encountered during the course of the project.

Appendix B

COVID 19

Being an exclusively computational project, COVID 19 had little to no impact on the ability to carry out project work. Disruptions came in the form of a lack of in-person supervisions. Consequently, supervisor-supervise meetings were conducted over virtual channels, thus mitigating any potential disruptions.