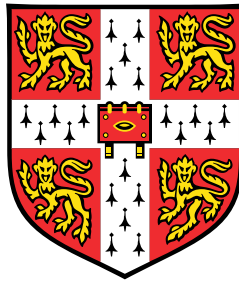


Automatic Assessment of English as a Second Language



Potsawee Manakul

Supervisor: Prof. Mark Gales

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Engineering

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation 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 dissertation 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 dissertation contains fewer than 12,000 words including appendices, bibliography, footnotes, tables and equations.

Potsawee Manakul

May 2019

Acknowledgements

I would like to thank Prof. Mark Gales for his guidance and support throughout the project, and for inspiring my interest in the area of spoken language processing. I also would like to thank Dr. Kate Knill for her valuable feedback and suggestion, and all her work on data processing. I would like to thank Dr. Anton Ragni and Dr. Yu Wang for providing technical support. Lastly, thank you all the ALTA members and my friends who gave me great support throughout my fourth year.

Abstract

Automatic language assessment and learning systems are becoming more popular to support the increasing demand in English language learning. One essential aspect of the automatic systems is to provide reliable and meaningful feedback to learners on grammatical errors they make. The learners can use the feedback to improve their language proficiency. The automatic systems can also provide more quality control to assessment processes. Working with spoken language provides further challenges due to its nature. Therefore, this work focuses on developing grammatical error detection and correction systems for spoken language.

To provide feedback, the problem of detecting grammatical errors in non-native spoken English is considered as a first step. The current state-of-the-art grammatical error detection (GED) systems are designed for written texts; as a result, they do not yield desirable performance on spoken language data. In this work, we are interested in adapting these systems to spoken language data. Firstly, we illustrate how a state-of-the-art GED system performs on both written and spoken data. Then, we provide a method of improving its performance by doing a system combination, and we compare this method to transfer learning. Also, this work aims to provide baseline GED results on a spoken language test set as a reference for future work.

The next step towards automatic systems is grammatical error correction (GEC) in non-native spoken English. We approach this problem by two methods: language model (LM) based approach, and neural machine translation (NMT) based approach. The first approach, LM-based, is motivated by the fact that it does not rely on annotated data, which is generally labourious to obtain. This approach involves generating a confusion set using the Automatically Generated Inflection Database (AGID) as well as pre-defined sets, and expanding the confusion set into a lattice for rescoring by a language model. Firstly, without using annotated data, we train an RNN language model on the one-billion corpus and achieve the $F_{0.5}$ score of 26.5 and 18.4 on the CoNLL and BULATS data sets respectively. Then, we make use of a GED system in confusion set generation by deriving candidates only for words predicted as incorrect, giving an improvement of around 3.2 in $F_{0.5}$. Next, we compare three variants of language models: n-gram, recurrent neural network (RNN), and succeeding word RNN (suRNN). It is found that the suRNNLM performs the best on both written on spoken test sets, outperforming the RNNLM by 1.0 and 0.4 in $F_{0.5}$ on CoNLL and BULATS respectively. Since applying GED shows a good improvement, we investigate the maximum gain we could achieve by an idealised GED system. Given known labels, this idea is experimented, and the gain in $F_{0.5}$ is found to be around 10

and 15 on CoNLL and BULATS respectively. As the correction candidates are derived from the AGID, the database limits the errors the system can correct, and it is found that the coverage on CoNLL is about 51.2% and that on BULATS is about 56.7%. Lastly, by tuning the bias of the original words in a lattice, the gain in $F_{0.5}$ on CoNLL is around 1.0, but there is almost no gain when evaluated on BULATS.

The second approach to GEC is neural machine translation (NMT). It is motivated by the recent success in deep learning on several machine translation and natural language processing tasks. Our NMT system is based on the bi-directional LSTM encoder-decoder architecture. When trained on sentences in the CLC and their corrected pairs, the NMT system achieves the $F_{0.5}$ score of 32.6 on CoNLL and 33.9 on BULATS, outperforming the LM-based with GED system on both evaluation sets. Moreover, when annotated data in the target domain are available, the NMT system can be fine-tuned, and it is shown that this technique gives a gain of 4.8 on CoNLL and a gain of 7.2 on BULATS. In summary, the findings recommend a practical guide towards building a GEC system as follows: (1) when there are no annotated data, an LM-based system using an suRNLM can achieve the highest performance, (2) if error tagged data are available, a GED system can be trained and used in confusion set generation to improve the overall performance of the LM-based GEC system, (3) if error corrected sentence pairs are available, an end-to-end NMT system achieves a higher performance level than the LM-based system, (4) if the target domain data are annotated for GEC, the NMT system can be fine-tuned to achieve the best performance level.

In addition, language models can be used to discriminate between more natural sentences and less natural sentences. We refer to the score of language model as the fluency score, which can be used for selecting the results of data augmentation, GEC system combination, or NMT-based GEC output re-ordering. It is shown that the quantity of training data is more important than the type of language model to obtain a good fluency score. The language model trained on the one-billion corpus gives the best result in our experiment. When used for the GEC system combination, we achieve a higher $F_{0.5}$ score compared to a single system.

Data augmentation is investigated as there are abundant speech data that are not annotated for grammatical errors. Our error generation models are based on n-gram and NMT. We show that augmented data from any model improve the GED performance. When comparing the models, the results suggest that the bigram and NMT with sampling decoding models are more suitable than the other models investigated. Moreover, augmenting data that are closer to the target corpus gives a better GED result.

Table of contents

List of figures	viii
List of tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Spoken Language vs Written Language	2
1.3 Outline of the report	2
2 Deep Learning	3
2.1 Overview	3
2.2 Feedforward Neural Networks	4
2.3 Recurrent Neural Networks	4
2.3.1 Bi-directional RNNs	5
2.3.2 Long Short Term Memory	5
2.4 Attention Mechanism	6
2.5 Transfer learning	7
3 Data	8
3.1 Corpora	8
3.1.1 Cambridge Learner Corpus (CLC)	9
3.1.2 BULATS	9
3.1.3 NICT-JLE	10
3.1.4 CoNLL	10
3.1.5 Native Speech Corpora	10
3.1.6 One-Billion	10
3.2 Data Processing	11
4 Grammatical Error Detection (GED)	12
4.1 Model Architecture	12
4.2 Evaluation Metrics	13
4.3 Experiments	14

4.3.1	GED system on written vs spoken language data	14
4.3.2	GED system combination	15
5	Grammatical Error Correction (GEC)	18
5.1	Language Model (LM) based Approach	18
5.1.1	Confusion Set Generation	19
5.1.2	Lattice Generation & Rescoring	20
5.1.3	Language Model	21
5.2	Neural Machine Translation (NMT) based Approach	22
5.2.1	Model Architecture	22
5.2.2	Implementation	24
5.3	Evaluation Metrics	25
5.4	Experiments	25
5.4.1	LM-based GEC	25
5.4.2	NMT-based GEC	29
5.5	Comparison between the two approaches	31
6	Fluency Score	33
6.1	The distribution of the Fluency Score on the CLC	33
6.2	GEC System Combination	35
6.2.1	Experiments	36
6.3	NMT-based system Re-ordering	37
6.3.1	Experiments	37
6.4	Summary	37
7	Data Augmentation	38
7.1	N-gram Statistics	38
7.2	NMT Statistics	40
7.3	Experiments	40
7.3.1	Corrupting Native Speech Corpora	40
7.3.2	GED with Data Augmentation	41
8	Conclusions and Future Work	43
	References	45
	Appendix A Risk Assessment	49

List of figures

2.1	An unrolled RNN unit	5
2.2	LSTM structure	6
4.1	Model Architecture of the GED system	13
4.2	Precision-Recall curves of the baseline GED system evaluated on FCE-public (written) and BULATS (spoken)	15
4.3	Precision-Recall curves of a set of GED models trained on the CLC and evaluated on FCE-public	16
4.4	Precision-recall curves of single written-text based, single speech based, and combined GED systems evaluated on BULATS	17
5.1	The pipeline of LM-based GEC	18
5.2	An example of confusion set and word lattice of the candidates	20
5.3	Standard encoder decoder with attention mechanism architecture. Figure adapted from the TensorFlow NMT tutorial: https://github.com/tensorflow/nmt	24
5.4	$F_{0.5}$ of LM-based (RNN1) system at different values of bias term	26
5.5	$F_{0.5}$ of LM-based (RNN1) system with GED at different values of bias term	27
5.6	LM-based (RNN1) GEC system at various GED's operating points.	27
5.7	Summary of the GEC systems where LM-based system uses RNN1 language model, NMT1 is the bi-LSTM model using beam search decoding, dropout, and scheduled sampling. TL denotes transfer learning.	32
6.1	The distribution of the difference of the log-probability	35
7.1	$F_{0.5}$ for the three experiments as the augmented data increase. Note that the results for settings 4 and 5 follow the same trend as settings 2 and 3.	42

List of tables

3.1	Summary of the <i>spoken</i> language corpora	8
3.2	Summary of the <i>written</i> language corpora	9
4.1	Precision, Recall, $F_{0.5}$ scores of the baseline GED system	14
4.2	Precision, Recall, and $F_{0.5}$ scores of the Text and Speech GED systems evaluated on BULATS	17
5.1	LM-based (RNN1) GEC performance on CoNLL and BULATS	26
5.2	LM-based GEC (with GED) with different LMs trained on the one-billion corpus	28
5.3	$F_{0.5}$ of LM-based using trained GED (denoted by GED) and with idealised GED (denoted by GED*)	28
5.4	Coverage of AGID on different data sets	29
5.5	GEC performance of NMT systems	30
5.6	Training Deeper Models	30
5.7	NMT system with transfer learning on CoNLL and BULATS data sets	31
5.8	Summary of data required to train different GEC systems. Note that corrected data refers to parallel texts consisting of the corrected version and the erroneous version. TL denotes transfer learning.	32
6.1	Fluency Score results based on 5-gram LM trained on different datasets	34
6.2	Fluency Score results based on different LMs trained on the one-billion corpus with vocabulary size of 64.0K	34
6.3	GEC system combination evaluated on CoNLL	36
6.4	GEC system combination evaluated on BULATS	36
6.5	Re-ordering the output of NMT evaluated on CoNLL & BULATS	37
7.1	Statistics of the CLC, BULATS, and native-speech corpora after corrupting using the modified unigram statistics.	41
7.2	The highest values of $F_{0.5}$ for different corrupting methods after folding corrupted data 10 times to FCE-public. The statistics is extracted from the CLC for all experiments. Baseline systems were trained on FCE-public without additional data.	42

Chapter 1

Introduction

1.1 Motivation

According to the British Council's research, over one billion people are learning English around the world, and millions of these take assessments to demonstrate their proficiency. Because of the high and rising demand, there is a growing shortage of qualified teachers and assessors in many countries. In the age of increasing availability of computers and online platforms, the development of speech processing and machine learning techniques for computer-assisted language learning (CALL) systems and automatic assessment systems could help learners improve their English more effectively, as well as automating the expensive and time-consuming assessment process.

One crucial aspect of these systems is to provide reliable and meaningful feedback to learners on errors they make. Language learners can use the feedback independently, or under the supervision of a teacher, to improve their proficiency. Another important aspect of these systems is that they could provide additional layers of quality control and speed up assessment processes. For computers, working with spoken language is even more difficult than written language as systems have to handle spontaneous speech from non-native speakers of different accents before they can assess and give feedback. One important aspect of assessment and feedback is English learners' choice of grammar which is challenging due to the nature of spoken language that when we speak, we hesitate, make repetitions, do not make complete sentences, make use of gesture and intonation. The performance of assessment and feedback systems also depends on automatic speech recognition (ASR) systems.

This project aims to investigate the extent to which machine learning and deep learning techniques can be applied to develop automatic systems, which consist of grammatical error detection and correction systems in order to assess proficiency and provide feedback.

1.2 Spoken Language vs Written Language

Working with spoken language data such as manual transcriptions of speech, or the outputs of ASR systems further complicates this task. Firstly, spoken language consists of pronunciation, prosody, and delivery in addition to text; as a result, messages are also delivered by body languages making sentences in spoken language less complete compared to written text. Secondly, when we speak we may not speak in full sentences, we make repetitions, we hesitate, for example. Thirdly, it is argued that there is no defined set of spoken grammar. Nevertheless, there are advantages of speech compared to written text. For instance, there are no spelling or pronunciation mistakes, and there is additional information in the audio signal such as confidence score which describes how likely that a word is correctly transcribed. The following example demonstrates the difference between speech and written text as well as grammatical errors:

- **Speech (Manual transcription or ASR output):**

hi peter um do you want to play tennis uh in evening yeah i um i want to play but i have to finishing the work you know can we do tomorrow

- **Meta-Data Extraction:**

Speaker1: / hi peter / **um** do you want to play tennis **uh** in evening /

Speaker2: / **yeah i um** i want to play / but i have to finishing the work **you know** / can we do tomorrow /

- **Written Text:**

Speaker1: Hi Peter, do you want to play tennis in evening?

Speaker2: I want to play, but I have to finishing the work. Can we do tomorrow?

- **Grammatical Errors:**

Speaker1: Hi Peter, do you want to play tennis in **[the]** evening?

Speaker2: I want to play, but I have to **[finishing → finish]** the work. Can we do tomorrow?

1.3 Outline of the report

The report consists of 7 chapters. Chapter 2 provides background knowledge on deep learning techniques used in this project. Chapter 3 describes data sets and pre-processing steps on the data. Chapter 4 describes the work on Grammatical Error Detection (GED). Chapter 5 describes the work on Grammatical Error Correction (GEC) including two approaches investigated: language model based approach and neural machine translation based approach. Chapter 6 examines the use of language models for further improvements of GEC systems by system combination and NMT re-ordering. Chapter 7 describes the methods for data augmentation, and the use of augmented data to train GED systems. Lastly, Chapter 8 concludes the main findings of this project and discusses future work.

Chapter 2

Deep Learning

2.1 Overview

Deep learning is a machine learning method that uses multiple layers of artificial neurons to progressively extract higher level features from raw input such as text. In recent years, deep learning architectures such as deep neural networks (DNN) and recurrent neural networks (RNN) have shown success in many areas such as speech processing, language modelling, and natural processing. This project applies deep learning to two types of machine learning tasks:

- **Sequence Labelling** - This task involves assigning a categorical label to each member of a sequence of observed values. Grammatical Error Detection (GED) is a sequence labelling task where each word in a sentence is assigned to one of the two classes: grammatically correct (c) or grammatically incorrect (i).
- **Sequence to Sequence** - This task involves generating an output sequence from an input sequence, and these two sequences may have different lengths. For instance, the machine translation problem is to translate a sentence from one language to another. Grammatical Error Correction (GEC) can be treated as a sequence to sequence problem where the input sentence is the erroneous sentence, and the output is a corrected sentence.

For both sequence labelling and sequence-to-sequence tasks, recurrent neural networks can be applied since they can incorporate the information from the entire input sequence in order to make a prediction. This chapter aims to provide background knowledge about deep learning techniques used in this project as follows:

- Feedforward Neural Networks (FFNNs): The simplest build block for deep learning.
- Recurrent Neural Networks (RNNs): The building block for our systems.
- Attention Mechanism: A crucial element for sequence-to-sequence models
- Transfer Learning: A method for reusing a trained model on a new problem.

2.2 Feedforward Neural Networks

Feedforward neural networks (FFNNs) are the first and simplest type of neural networks. The connections between the nodes do not form a cycle as opposed to recurrent neural networks. The network could be single-layer perceptron which consists of a single layer of output nodes, or multi-layer perceptron having multiple layers of computational units. In this work, a *fully-connected* connection is used between layers meaning that every input node is connected to every node in the next layer. For each layer, the network computes the output \mathbf{y} given the input \mathbf{x} by:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (2.1)$$

where σ is a non-linear activation function such as tanh, ReLU, sigmoid, or softmax. Note that softmax is usually used at the final layer as it takes a vector K real numbers as input, and normalises it into a probability distribution:

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)} \quad (2.2)$$

2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) have internal or hidden state (memory) for processing sequences of inputs. At time step t , a recurrent unit computes the hidden state \mathbf{h}_t as a function of the current input \mathbf{x}_t and its previous hidden state \mathbf{h}_{t-1} , and the output at this time step is computed as a function of the hidden state:

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\mathbf{h}_{t-1} + \mathbf{b}_h) \quad (2.3)$$

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_y) \quad (2.4)$$

where σ_h and σ_y are non-linear activation functions, and \mathbf{W}_h , \mathbf{U}_h , \mathbf{b}_h , \mathbf{W}_y , and \mathbf{b}_y are the model parameters (weights) to be trained. An unrolled recurrent unit can be view as a deep fully connected neural network except that the weights are shared. Training the RNN is done by backpropagation through time (BPTT) [1] rather than the standard error backpropagation for feed-forward neural networks. Multiple recurrent units can be stacked together to form a deep RNN architecture.

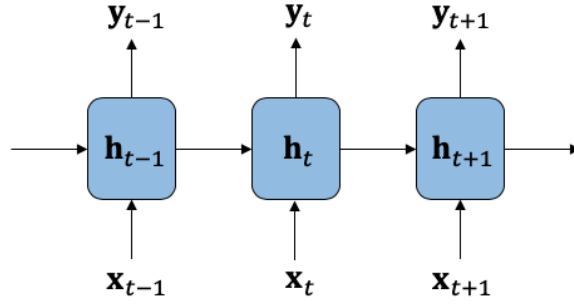


Fig. 2.1 An unrolled RNN unit

2.3.1 Bi-directional RNNs

Two recurrent units can be used to incorporate both past and future information. Each unit has its own weights, and the output is a function of both hidden states:

$$\mathbf{h}_t^{(f)} = \text{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}^{(f)}) \quad (2.5)$$

$$\mathbf{h}_t^{(b)} = \text{RNN}(\mathbf{x}_t, \mathbf{h}_{t+1}^{(b)}) \quad (2.6)$$

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_y^{(f)} \mathbf{h}_t^{(f)} + \mathbf{W}_y^{(b)} \mathbf{h}_t^{(b)} + \mathbf{b}_y) \quad (2.7)$$

This network is called bi-directional RNN [2]. When it is used to encode a sequence into a single representation vector the final hidden state is used for the forward unit, and the first hidden state is used for the backward unit.

2.3.2 Long Short Term Memory

Long Short Term Memory (LSTM) [3] is an RNN architecture, which is a modification of the vanilla RNN to handle long term dependencies. The vanilla RNN suffers the gradient vanishing or exploding problem when performing error back-propagation. LSTM deals with this problem by using an input gate \mathbf{i}_t , an output gate \mathbf{o}_t , and a forget gate \mathbf{f}_t . The operation can be written as:

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \quad (2.8)$$

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \quad (2.9)$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \quad (2.10)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \sigma_c(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \quad (2.11)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \sigma_h(\mathbf{c}_t) \quad (2.12)$$

where σ_g , σ_c , and σ_h are point-wise non-linear activation functions.

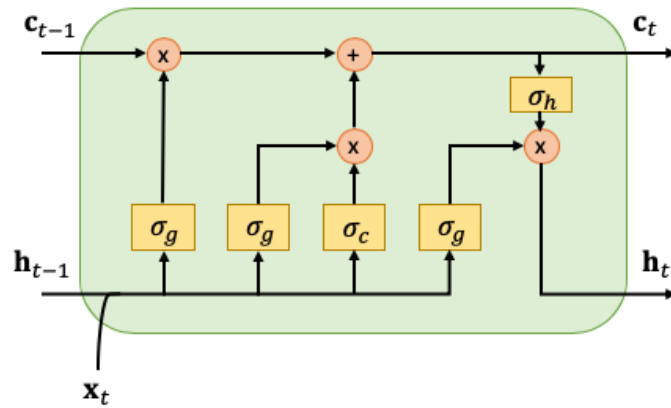


Fig. 2.2 LSTM structure

2.4 Attention Mechanism

Attention mechanism is important because it has been shown to produce state-of-the-art results in machine translation and other natural language processing tasks [4, 5]. When it is applied to an RNN, it allows the RNN to encode a sequence better by focusing on parts of hidden states instead of using the final hidden state. Given hidden states of an RNN, $\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_T$, the attention mechanism computes the following:

- Score vector:

$$e_t = \text{score}(\mathbf{k}, \mathbf{h}_t) \quad (2.13)$$

- Attention weights:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{i=1}^T \exp(e_i)} \quad (2.14)$$

- Context vector:

$$\mathbf{c} = \sum_{t=1}^T \alpha_t \mathbf{h}_t \quad (2.15)$$

where the score function has two standard forms: *dot-product* attention $e_t = \mathbf{h}_t^T \mathbf{W}_{xk} \mathbf{k}$, and *additive* attention $e_t = \mathbf{w}^T \tanh(\mathbf{W}_x \mathbf{h}_t + \mathbf{W}_k \mathbf{k}_t)$. The key \mathbf{k} can be chosen as the input vector \mathbf{x}_s resulting in e_{st} , α_{st} , and \mathbf{c}_s ; this form of attention mechanism is called *self-attention* mechanism.

2.5 Transfer learning

Transfer learning is the process of storing knowledge gained from solving one problem and apply it to another problem which is different but related. Transfer learning is popular in deep learning due to a large amount of data required to train deep learning models. A common approach involves:

1. Choose a related predictive modelling task with a large amount of training data and develop a model for the chosen task. Alternatively, for many common tasks, a model can be selected from a range of available pre-trained models.
2. Reuse the trained model either entirely or partially on the target task.
3. Fine-tune the model on the input-output pair data available to the target task.

For example, in the field of natural language processing (NLP), pre-trained language representation models are used on a wide range of NLP tasks, many of which achieve state-of-the-art results [6–8]. Due to the lack of large speech corpora with grammatical error mark-up, transfer learning can be used to boost the performance of automatic systems for GED and GEC problems by training a deep learning model on large written-text data, and re-using it as a pre-trained model to fine-tune it on the target speech data set.

Chapter 3

Data

This chapter aims to provide an overview of various data sets used in the project. The main aspects of the data sets for this project are:

- **Style:** *Written* or *Spoken* - This aspect describes the nature of the data, whether they are written text or transcripts of speech. The transcription could be a manual transcription by a human or the output of an automatic speech recognition (ASR) system.
- **Annotation:** *Unannotated*, *Error Tagged*, or *Corrected* - Unannotated data are plain texts which are, in general, relatively easy and cheap to obtain. Error tagged data are for the GED task as each word is labelled with either being grammatically correct (c) or incorrect (i). Corrected data are parallel texts: one is produced by English learners, and the other is the corrected version.

3.1 Corpora

Corpus	Annotation	Usage	# Words
AMI	Unannotated	Training	2.0M
Switchboard	Unannotated	Training	3.1M
MGB	Unannotated	Training	3.8M
Fisher	Unannotated	Training	35.0M
BULATS	Error Tagged & Corrected	Evaluation	61.9K
NICT-JLE	Error Tagged & Corrected	Evaluation	135.3K

Table 3.1 Summary of the *spoken* language corpora

Corpus	Style	Usage	# Words
One-Billion	Unannotated	Training	730M
CLC	Error Tagged & Corrected	Training	14.1M
FCE-public	Error Tagged & Corrected	Training/Evaluation	450K
CoNLL	Corrected	Evaluation	27.8K

Table 3.2 Summary of the *written* language corpora

3.1.1 Cambridge Learner Corpus (CLC)

The Cambridge Learner Corpus [9] is a fully annotated corpus of English learners. In total, it contains over 23 million words of text from examination scripts written by candidates taking written tests from Cambridge Assessment. It has been manually annotated with grammatical errors, and the errors have also been corrected. The subset FCE-public, which is publicly available, consists of 1,244 exam scripts of candidates taking FCE examinations in 2000 and 20001; this publicly available data set is also divided into train, dev, and test sets. In this project, the CLC for training includes the FCE-public training set, and written tests from IELTS, BULATS, CPE, and CAE giving approximately 14 million words or 870 thousand sentences from 27.5 thousand of candidates. In addition, other examinations, including BEC, ICFE, ILEC, KET, SFL, and PET, are used in Chapter 6.

3.1.2 BULATS

The corpus is the main speech data used as an evaluation set in this project. It is from the Business Language Testing Service (BULATS) test provided by Cambridge Assessment. The test consists of read aloud and free-speaking parts. In this project, only the free speaking C, D, and E sections are used. The data contain 226 candidates. In section C, the candidates are given a business or work-related topic to talk about for 60 seconds. In section D, charts and/or graphs are provided, and the candidates are asked to describe them for 60 seconds. In section E, the candidates are asked to answer five questions related to one scenario. The data set consists of 1438 responses from these sections. Speakers are approximately evenly distributed across CEFR grade A1-C (C1 and C2 are combined). They come from 6 first languages (L1s): Arabic, Dutch, French, Polish, Thai, and Vietnamese. Two sets of manual transcription with grammatical error markup and additional meta-data annotations have been produced [10]. Only one annotation is used in this project. In addition to manual transcriptions, ASR transcriptions of the BULATS data were also made based on the graphemic stacked hybrid DNN+LSTM-HMM joint decoding system described in [11].

3.1.3 NICT-JLE

The NICE Japanese Learner English (JLE) Corpus (v4.1) [12] consists of manual transcriptions from a speaking English test ACTFL-ALC SST. A native or proficient speaker of English conducted an interview with a candidate. The corpus contains 167 interviews which have been annotated with grammatical errors and disfluencies. All candidates are Japanese L1 speakers. Only the candidate parts are used in this project. The distribution of scores is equivalent to A1-B2 on the CEFR scale.

3.1.4 CoNLL

This is the official *test data* from the CoNLL-2014 Shared Task: Grammatical Error Correction [13]. The test set was collected from 25 students, who are non-native speakers of English, from the National University of Singapore. They were recruited to write essays in response to prompts. The data consist of 1.3 thousand sentences that were annotated for grammatical error correction by two experts, independently giving two sets of annotation. The CoNLL test data are processed further in this project, including the removal of punctuations and correcting spelling mistakes as motivated by the nature of speech data. Since the data were made for the GEC task, there are no grammatical error labels. In this work, the labels are obtained from the locations of correction.

3.1.5 Native Speech Corpora

The corpora in this category comprise: (1) AMI [14] - manual transcriptions of meeting recordings, containing about 2.0 million words, (2) Switchboard [15] - manual transcriptions of telephone speech, containing about 3.1 million words, (3) Multi-Genre Broadcast (MGB) ¹ - manual transcriptions of multi-genre broadcast, containing about 3.8 million words, and (4) Fisher [16] - manual transcriptions of telephone speech, containing approximately 35 million words. In this project, the assumption for using these data sets is that native speakers of English produced the recordings; hence, their choice of grammar is assumed correct.

3.1.6 One-Billion

The one-billion corpus [17] is widely used as a benchmark in language modelling. The data were collected from resources that are freely available on the internet. This corpus is by far the largest available to this project with the size about 730 million words after processing. It does not contain any information or annotation about grammatical errors.

¹<http://www.mgb-challenge.org/>

3.2 Data Processing

In this work, spelling mistakes and punctuation are removed from all the written text corpora such as the CLC and CoNLL. The intuition is that our interest is to build systems for speech data, and there are no spelling mistakes or punctuation in speech. The texts are lowercased such that they are as close to the ASR output as possible. Furthermore, further processing steps are carried out for each of the tasks.

- **Pre-processing data for GED**

- If there is a missing word, the following word is labelled as incorrect. The intuition is that although this word is correct when considered in isolation, it is incorrect given the context.
- `</s>` token is prepended and appended to each sentence to add sentence boundaries. Note that `<s>` is not included in the pre-trained Google's 3 million word word2vec word embedding, so `</s>` is used for both sentence start and end.
- In speech corpora, repetitions, hesitations, and partial words are removed.
- Example of a sentence with labels:

```
sentence:  we  can  inform  with  customers
label:    c   c    c      i      c
```

- **Pre-processing data for GEC**

- Sentences are tokenised using the Natural Language Toolkit (NLTK) [18]
- Sentence start `<s>` and sentence end `</s>` tokens are used.
- Example of a pair of sentences:

```
incorrect: internet was something amazing for me
correct:  the internet was something amazing for me
```

Chapter 4

Grammatical Error Detection (GED)

This chapter considers the problem of detecting *grammatical* errors in non-native spoken English as a first step towards automatic language assessment and learning systems. More specially, the task is to develop a system that labels each of the words in a given sentence as being grammatically correct (c) or incorrect (i). We investigate the use of deep learning for building a GED system. The GED system is initially developed for written texts, and its performance on a manual transcription of speech is investigated. Furthermore, two methods for improving the GED system’s performance on spoken language data are presented.

4.1 Model Architecture

A state-of-the-art neural network based grammatical error detection (GED) system is chosen for this work. It is designed to detect all types of grammatical errors present in texts. This GED system is the sequence labeller¹ having a bi-directional LSTM network architecture, assigns the probability of each word being grammatically incorrect [19]. The GED system, firstly, splits an input sentence into a sequence of tokens $w_{1:N} = \{w_1, \dots, w_N\}$. Using a word embedding, the tokens are mapped to vector representations. Each of the word-level vector representations is concatenated with a character-level vector representing that word obtained by a bi-directional LSTM layer as described in [20]. These combined vectors, $\mathbf{x}_{1:N} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, are passed to another bi-directional LSTM layer which encodes $\mathbf{x}_{1:N}$ into context-specific forward and backward representations $\vec{\mathbf{h}}_{1:N}$ and $\overleftarrow{\mathbf{h}}_{1:N}$.

$$\vec{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \vec{\mathbf{h}}_{t-1}); \quad \overleftarrow{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \overleftarrow{\mathbf{h}}_{t+1}) \quad (4.1)$$

The forward and backward vectors are then concatenated into a hidden vector \mathbf{d}_t . Subsequently, the hidden vector is passed to a feedforward layer with tanh activation function, and the output probability is obtained from another feedforward layer having two neurons and the softmax

¹<https://github.com/marekrei/sequence-labeler>

function.

$$\mathbf{d}_t = \text{concat}(\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t) \quad (4.2)$$

$$P(\mathbf{y}_t | w_{1:N}) = \text{softmax}(\mathbf{W}_o(\tanh(\mathbf{W}_d \mathbf{d}_t))) \quad (4.3)$$

The trainable parameters are the embedding matrix, bi-directional LSTM weights, and feedforward weights, and the cost function being minimised in training is the cross-entropy between output probability and the annotated labels.

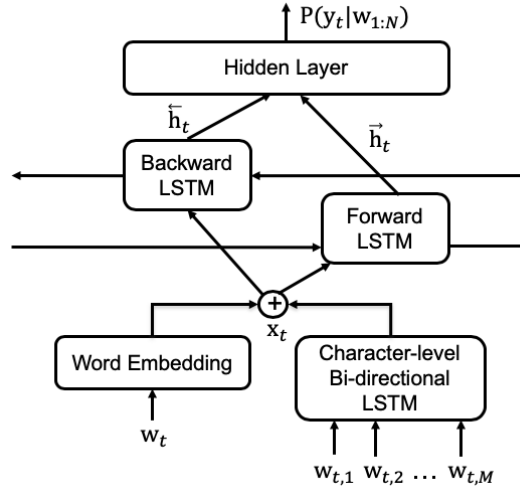


Fig. 4.1 Model Architecture of the GED system

In this project, the word embedding is 300-dimensional, and the character-level embedding vector is 50-dimensional (\mathbf{x}_t is 350-dimensional). Both forward and backward LSTM layers have 200 units ($\vec{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$ are 200-dimensional; hence, \mathbf{d}_t is 400-dimensional), and the hidden layer has 50 units.

4.2 Evaluation Metrics

Evaluation is performed by computing precision (P), recall (R), and F-score:

$$P = \frac{TP}{TP + FP} \quad (4.4)$$

$$R = \frac{TP}{TP + FN} \quad (4.5)$$

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \quad (4.6)$$

where TP (true positive) is the number of tokens correctly labelled as incorrect (i), FP (false positive) is the number of tokens labelled as incorrect (i) but in fact are correct, and FN (false negative) is the number of tokens labelled as correct (c), but in fact are incorrect. The main

evaluation measure for error detection commonly adopted for GED is $F_{0.5}$ which combines both precision and recall, while emphasises precision as twice as recall since accurate/reliable feedback is more important than coverage in the error detection task [21].

4.3 Experiments

4.3.1 GED system on written vs spoken language data

Since the CLC is the largest corpus available with grammatical error labels, it is chosen for training the GED system, which will be referred to as the *baseline* system in this experiment. The baseline system has an $F_{0.5}$ score of 57.6 as shown in Table 4.1. This performance level is lower than the results reported in [19, 22] since the spelling and punctuation mistakes, which account for 23% in the FCE-public test set, have been corrected during pre-processing. Figure 4.2 shows that the precision on BULATS is lower than that of FCE-public at all recall rates. This is due to the fact that the FCE-public data match the training data (the CLC), but the BULATS data, which are spoken language, have different word distributions.

Figure 4.2 suggests that even at the low recall region, the system has only 80% precision rate on BULATS. Since the objective is to build a GED system suitable for speech data, we need to improve the system to provide meaningful feedback. In [23], such a system is enhanced by a fine-tuning method giving an $F_{0.5}$ score of 55.8 evaluated on BULATS. This result is promising, but the performance is still worse than when it is performed on written data. In Section 4.3.2, it investigates an alternative method to improve the performance by doing system combination, and in Chapter 7 data augmentation approach will be examined in detail.

Test Set	P	R	$F_{0.5}$
FCE-public	69.9	33.9	57.6
BULATS	52.4	27.0	44.1

Table 4.1 Precision, Recall, $F_{0.5}$ scores of the baseline GED system

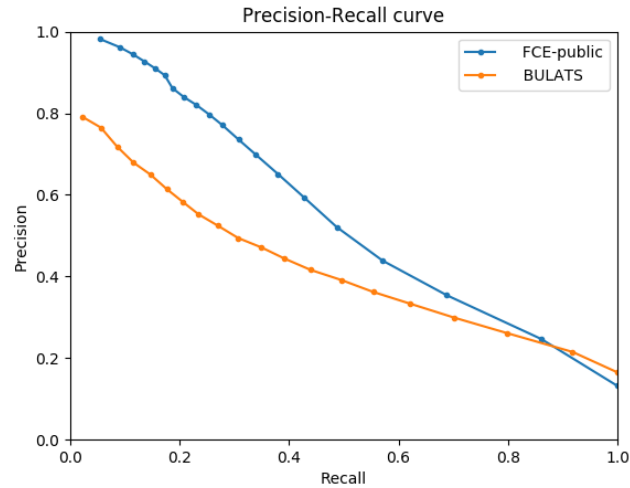


Fig. 4.2 Precision-Recall curves of the baseline GED system evaluated on FCE-public (written) and BULATS (spoken)

4.3.2 GED system combination

In machine learning, particularly neural network based methods, **ensemble averaging** is the process of training multiple models and combining them to produce the desired output rather than training one model. Empirically, an ensemble of models performs better than any individual model, because the various errors of models *average out*. One way of obtaining an ensemble is to train a set of models with different topologies, or we could train a set of models with the same topology but different training configurations such as learning rate or random seed. So, an ensemble of GED models, $\mathcal{E} = \{M^1, \dots, M^J\}$, makes a prediction by:

$$P(\mathbf{y}_t | w_{1:N}; \mathcal{E}) = \frac{1}{N} \sum_{i=1}^J P(\mathbf{y}_t | w_{1:N}; M^{(i)}) \quad (4.7)$$

Firstly, we investigate the use of ensemble by varying the randomness, i.e. *random seed* in Python's numpy and TensorFlow packages. Five models are trained on the CLC, and the result is shown in Figure 4.3. Overall, the ensemble average method has a small gain over a single model; the $F_{0.5}$ score increases by **0.68** from the average of the $F_{0.5}$ scores of the five individual systems.

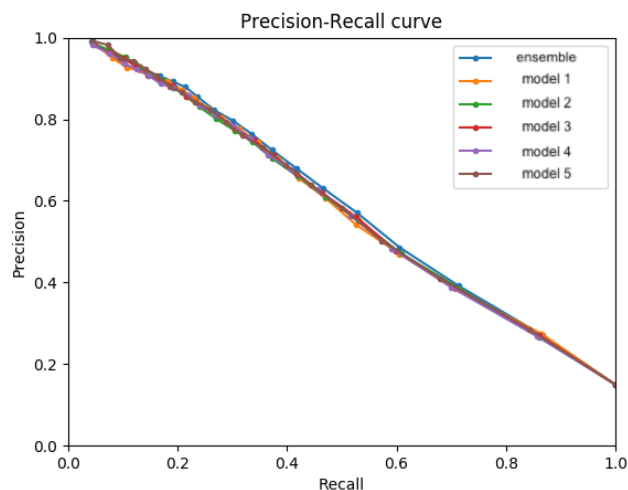


Fig. 4.3 Precision-Recall curves of a set of GED models trained on the CLC and evaluated on FCE-public

Secondly, we instead train a set of models with the same topology and the same training configuration on different data sets. This experiment makes use of the *speech* data sets such as AMI and Switchboard by corrupting them using the error statistics from the CLC; note that section 7 will describe how to corrupt native-speech corpora in details. This experiment uses the modified unigram statistics corruption method to obtain *speech* training data for GED systems. Once GED systems have been trained on different corpora, an ensemble of two systems, one trained on the CLC (text-CLC GED) and the other trained on speech data (speech GED), is investigated. The intuition is that false positive errors made by the text-CLC system due to the difference in the nature between written-text and speech will be averaged out by the speech GED system.

Table 4.2 shows that both combined systems, e.g. system 9 and system 10, have higher precision, recall, and $F_{0.5}$ than the baseline GED system, and we achieve a gain of **2.1** in $F_{0.5}$. Additionally, Figure 4.4 shows that the precision of the combined system is higher than the baseline system at all operating points.

System	P	R	F _{0.5}
(1) Text-CLC (baseline)	52.4	27.0	44.1
(2) Speech-AMI	39.1	26.5	35.7
(3) Speech-SWBD	45.0	23.4	38.0
(4) Speech-MGB	49.6	24.3	41.0
(5) Speech-Fisher	46.9	23.8	39.3
(6) Speech-AMI+SWBD	46.8	24.5	39.6
(7) Speech-AMI+MGB	48.5	26.5	41.6
(8) Speech-SWBD+MGB	48.5	27.5	42.1
(9) Combined-(1)&(6)	53.8	29.1	46.0
(10) Combined-(1)&(8)	53.5	29.9	46.2

Table 4.2 Precision, Recall, and F_{0.5} scores of the Text and Speech GED systems evaluated on BULATS

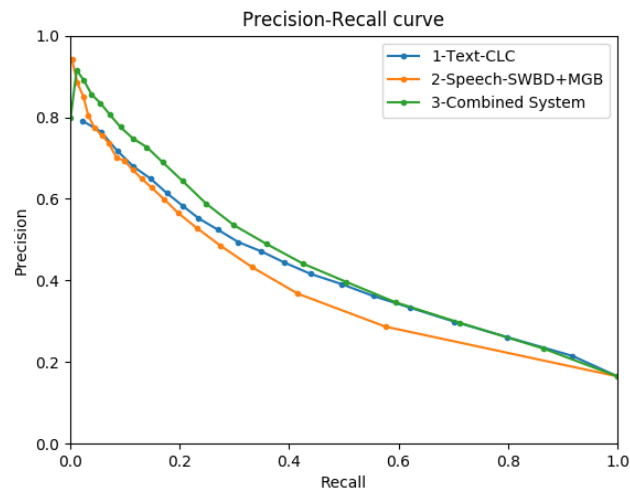


Fig. 4.4 Precision-recall curves of single written-text based, single speech based, and combined GED systems evaluated on BULATS

Chapter 5

Grammatical Error Correction (GEC)

This chapter considers the next step towards automatic language assessment and learning systems that is the problem of automatically correcting *grammatical* errors in non-native spoken English. This task is to develop a system that corrects grammatical errors in a given sentence. This chapter investigates two approaches to building a GEC system. First, a GEC system is based on a language model (LM) which is motivated by the fact that it does not require annotated data, which is generally expensive and time-consuming to obtain. Second, a GED system is based on a neural machine translation (NMT) model, which could capture more complex error patterns but requires parallel corpora (erroneous and error-corrected) to train. This chapter uses the processed CoNLL and BULATS data sets to evaluate the performance of both approaches and will conclude the chapter by comparing the two approaches.

5.1 Language Model (LM) based Approach

This approach comprises *two* main components: confusion set generation, and lattice generation & language model rescoring. In the first stage, a confusion set is constructed from the input sentence to represent possible hypotheses or correction candidates. A GED system may be used in this stage to reduce the number of hypotheses. In the second stage, the confusion set is converted into its lattice representation, and a language model is used to rescore the lattice in order to find the most probable sentence from the lattice, which will be the output of this system.

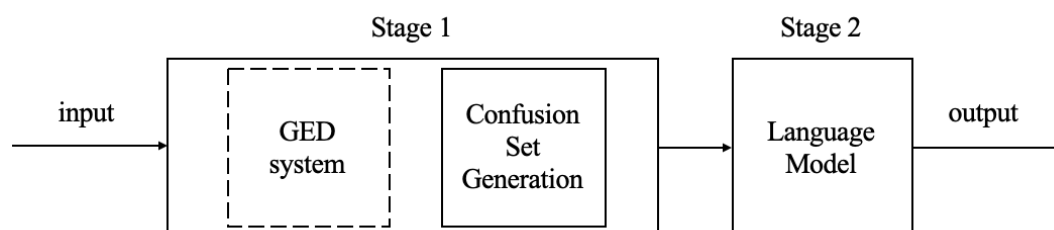


Fig. 5.1 The pipeline of LM-based GEC

5.1.1 Confusion Set Generation

A confusion set is an efficient representation of a set of possible candidate sentences. In the confusion set, the candidates of each word are represented by arcs. An example of confusion set is shown in Figure 5.2a. For each word, its correction candidates are derived from the Automatically Generated Inflection Database (AGID)¹. The AGID contains the inflected forms of a number of words in English. However, if the word is an article or a preposition, the candidates are obtained from a set of pre-defined words as follows:

- Prepositions: [ϕ , 'about', 'at', 'by', 'for', 'from', 'in', 'of', 'on', 'to', 'with']
- Articles: [ϕ , 'a', 'an', 'the']

where ϕ is the empty token meaning the word should be removed. Examples of correction candidates derived from the AGID are: bear \rightarrow [bear, bears, bore, born, bearing], tall \rightarrow [tall, taller, tallest]. Furthermore, if the part of speech of the word can be identified, the correction candidates can be narrowed down, e.g. bear (N) \rightarrow [bear, bears], or bear (V) \rightarrow [bear, bears, bore, born, bearing]. In addition, a GED system can reduce the number of words to be expanded. The GED system assigns the probability of each word being grammatically incorrect, and if this probability is lower than a threshold, e.g. 0.5, no candidates will be derived. It should be noted that although training a GED system requires data to have error tags, it does not require pairs of corrected and erroneous sentences to train. The steps are summarised in the pseudo-code below.

```

1: for word  $w_t$  in  $w_{1:N}$  do
2:   if  $P(y_t = \text{incorrect} | w_{1:N}) < \text{threshold}$  then                                 $\triangleright$  when GED is enabled
3:      $w_t^{1:K_t} \leftarrow w_t$                                                      $\triangleright$  when GED is enabled
4:     continue                                                                     $\triangleright$  when GED is enabled
5:   end if                                                                            $\triangleright$  when GED is enabled
6:   if  $w_t$  in AGID then
7:      $w_t^{1:K_t} \leftarrow \text{AGID}(w_t)$ 
8:   else if  $w_t$  in Prepositions then
9:      $w_t^{1:K_t} \leftarrow \text{Prepositions}$ 
10:  else if  $w_t$  in Articles then
11:     $w_t^{1:K_t} \leftarrow \text{Articles}$ 
12:  else
13:     $w_t^{1:K_t} \leftarrow w_t$ 
14:  end if
15: end for

```

¹<http://wordlist.aspell.net/other/>

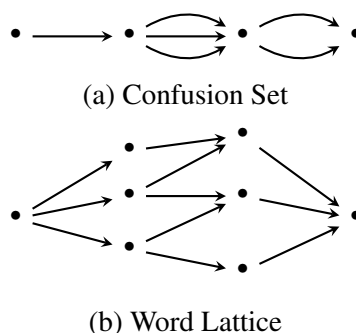


Fig. 5.2 An example of confusion set and word lattice of the candidates

5.1.2 Lattice Generation & Rescoring

Once the confusion set has been created, it is converted into word lattice, which is an efficient way of representing and decoding the list of candidate sentences which grow exponentially with the number of words in a sentence. The reason for this conversion is that a large language model will be used to decode, i.e. find the most likely path, efficiently. Since the Hidden Markov Model Toolkit (HTK) [24] is used for lattice generation and rescoring, the confusion set described in Section 5.1.1 is stored in the EBNF format. The conversion to word lattice is based on the HParse tool in the HTK² that convert the EBNF format to the HTK Standard Lattice Format (SLF). An example of a word lattice is shown in Figure 5.2b. There is related work on lattice generation; for instance, in [25], confusion sets are converted to word lattices using finite state transducer operations.

The lattice generation sets language model scores to zero as well as acoustic scores, which are irrelevant in this project. In addition, it should be reasonable to assume that most of the words in any sentence is more likely to be correct than incorrect. For example, the gold-standard annotation of the CoNLL shows that grammatical errors account for about 18%. In Section 7.3, the error statistics of the BULATS data is found to be 19.1%, and that of the CLC is 10.4%. This information provides the *prior* knowledge about the lattice that the words from the original sentence are more likely to be correct; hence, there is no correction required. We can, therefore, incorporate this *prior* into the lattice by **adding bias** to the arcs corresponding to such words since language model scoring is in the logarithmic domain.

In order to find the most likely sentence, a language model is used to *rescore* the lattice. The intuition is that the language model will give higher scores to sequences of words that appear more frequently in the training data assuming that the sentences in the training data, e.g. the one-billion corpus have no grammatical errors. Therefore, this approach should output the sequence that is the most natural or the most likely. The next section will provide background knowledge of the language models used.

²<http://htk.eng.cam.ac.uk/>

5.1.3 Language Model

A language model computes a probability for a sequence of words $p(w_1, \dots, w_N)$. In the final stage of the LM-based GEC pipeline, the language model is used to find the most grammatical sentence in a lattice, e.g. $p(\text{he likes playing tennis}) > p(\text{he likes play tennis})$:

$$\hat{w}_{1:N} = \underset{\forall w_{1:N}}{\operatorname{argmax}} p(w_{1:N}) = \underset{\forall w_{1:N}}{\operatorname{argmax}} \prod_{i=1}^N p(w_i | w_{i-1}, \dots, w_1) \quad (5.1)$$

This work investigates three types of language models as follows:

- **N-gram:** An n-gram model is a probabilistic language model that computes the probability of a word given its previous $(n - 1)$ words. During training, the probability of word w_i given words $w_{i-1}, w_{i-2}, \dots, w_{i-(n-1)}$ is modelled as:

$$p(w_i | w_{i-(n-1):(i-1)}) = \frac{C(w_{i-(n-1):(i-1)}, w_i)}{C(w_{i-(n-1):(i-1)})} \quad (5.2)$$

For instance, for the unigram and bigram models, these are:

$$p(w_2 | w_1) = \frac{C(w_1, w_2)}{C(w_1)}; \quad p(w_3 | w_1, w_2) = \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} \quad (5.3)$$

Subsequently, the n-gram model estimates the probability of a sequence $w_{1:N}$ by:

$$p(w_1, \dots, w_N) = \prod_{i=1}^N p(w_i | w_{1:(i-1)}) \approx \prod_{i=1}^N p(w_i | w_{i-(n-1):(i-1)}) \quad (5.4)$$

- **Recurrent Neural Network (RNN):** An RNN Language Model (RNNLM) computes the probability conditioned on *all* previous words by recursively multiplying weights at each time step meaning it uses a vector \mathbf{h}_t to represent the full history $\{w_1, \dots, w_{t-1}\}$. Given a list of word input vectors $\mathbf{x}_1, \dots, \mathbf{x}_N$, it computes:

$$\mathbf{h}_t = \operatorname{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}) \quad (5.5)$$

$$\hat{\mathbf{y}}_t = \operatorname{softmax}(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y) \quad (5.6)$$

$$p(w_{t+1} = \text{word}_j | w_t, \dots, w_1) \approx \hat{y}_{t,j} \quad (5.7)$$

The softmax layer has a normalisation term $\sum_{i=1}^{|V|} \exp(\mathbf{w}_{y,i}^T \mathbf{h}_t + b_{y,i})$ involving computing the size of vocabulary $|V|$ times, which is large, at each time step. This makes training RNNLMs difficult, and it requires estimation methods such as NCE. The effective training methods are not in the scope of this work, but can be found in [26, 27].

- **succeeding words Recurrent Neural Network (suRNN):** Since a complete sequence of words is obtained before grammatical error correction, future word context can be used. A

bi-directional RNNLM can incorporate all previous and future words; however, training it efficiently is still challenging and not investigated in this project. Instead, an suRNNLM that incorporates *some* future context and all previous context can be used [28]. The probability is computed as

$$p(w_1, \dots, w_N) = \frac{1}{Z} \prod_{i=1}^N p(w_i | w_{1:(i-1)}, w_{(i+1):N}) \quad (5.8)$$

Although Equation 5.8 requires the normalisation term Z , for the GEC task, only relative measures are required (to find the sentence with the highest probability). The advantage of this model over RNNLMs is that it better approximates unnormalised $p(w_1, \dots, w_N)$ by incorporating future context.

5.2 Neural Machine Translation (NMT) based Approach

Although the LM-based system can be trained without annotated data, it would require significant human work to handle complex errors such as word re-ordering or deletion errors. Another approach to GEC which can handle these errors better is to treat the task as a machine translation problem. Traditionally, machine translation was done by breaking up sentences into multiple parts and then translated them phrase-by-phrase. This system is also called *phrase-based* statistical machine translation (SMT). SMT was widely used for GEC, and several of the best performing systems in the CoNLL 2014 Shared Task on Grammatical Error Correction [13] were based on SMT [29–32]. More recently, a neural machine translation (NMT) system started to replace SMT because many error patterns are not captured by SMT systems due to data sparsity. Unlike SMT, an NMT system addresses the local translation problem by using RNN which can capture longer-term dependencies in sentences [33]. It reads the entire source sentence, encodes the sentence into a vector representation, and makes a translation based on this representation. This process of translation is more similar to how humans translate. As an NMT system translates a source sentence to its target sentence, we can apply it to GEC by treating an erroneous sentence as the source, and its correction as the target. There has been work which uses NMT models for GEC for written texts such as [34–36]. The following parts will describe the architecture of an NMT system used in this project.

5.2.1 Model Architecture

The NMT model is based on the encoder-decoder architecture. More specifically, the encoder uses bi-directional LSTM to build a vector representing the input sentence. Given an input sentence $\{w_1, \dots, w_N\}$, each word is mapped using a word embedding giving the input vectors

$\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$. These vectors are passed to the bi-directional LSTM encoder:

$$\vec{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \vec{\mathbf{h}}_{t-1}) \quad (5.9)$$

$$\overleftarrow{\mathbf{h}}_t = \text{LSTM}(\mathbf{x}_t, \overleftarrow{\mathbf{h}}_{t+1}) \quad (5.10)$$

$$\bar{\mathbf{h}}_t = \text{concat}(\vec{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t) \quad (5.11)$$

The decoder processes the encoded vector to make a translation. During *training*, the target sentence $\{w'_1, \dots, w'_M\}$ is available. They are mapped to $\{\mathbf{y}_1, \dots, \mathbf{y}_M\}$ using word embedding as in the encoder. The decoder has two uni-directional LSTMs since it makes translation in one direction.

$$\mathbf{h}_t^{(1)} = \text{LSTM}(\mathbf{y}_t, \mathbf{h}_{t-1}^{(1)}) \quad (5.12)$$

$$\mathbf{h}_t^{(2)} = \text{LSTM}(\mathbf{h}_t^{(1)}, \mathbf{h}_{t-1}^{(2)}) \quad (5.13)$$

and the first time step of the first LSTM is fed a special token $\mathbf{x}_{\langle \text{go} \rangle}$, and the final state of the encoder is used as an LSTM decoder input state. An attention mechanism with dot-product attention described in Section 2.4 is used to obtain a context vector for each decoder time step:

$$\mathbf{c}_t = \text{attention}(\text{key} = \mathbf{h}_t^{(2)}, \text{states} = \{\bar{\mathbf{h}}_1, \dots, \bar{\mathbf{h}}_N\}) \quad (5.14)$$

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

$$\mathbf{a}_t = \tanh(\mathbf{W}_c \mathbf{c}_t + \mathbf{W}_h \mathbf{h}_t^{(2)} + \mathbf{b}) \quad (5.15)$$

The output word at time step t is translated by finding the closest vector to \mathbf{a}_t in the word embedding. During *inference*, \mathbf{y}_t is not known, so \mathbf{a}_{t-1} is instead used, and this decoding method is called greedy decoding. There are additional techniques which are added to the model as follows:

- *Dropout* [37]: De-activate some nodes in the network randomly to prevent a single node from specialising to a task. This method is a way of regularisation.
- *Beam search decoding*: Keep n best sequences at all time steps during inference. Note that greedy decoding is equivalent to using the beam size of 1.
- *Scheduled sampling* [38]: During inference, the unknown previous token is replaced by a token generated by the model. This introduces the discrepancy between training and inference which results in errors that accumulate as it translates. This technique incrementally changes from target sequence being fully available to totally using predictions made by the model during training.

The multi-layer bi-directional LSTM architecture is also experimented. Since the bi-directional LSTM encoder has 2 independent LSTMs propagating information in different directions, the deeper network will have an even number of layers. For an n -layer NMT, $n/2$ LSTMs are stacked in the forward encoder, $n/2$ LSTMs are stacked in the backward encoder, and n LSTMs are stacked in the decoder. To train large neural networks efficiently, the residual connection [39] is applied between the stacked LSTMs:

$$\mathbf{h}_t^{(k+1)} = \text{LSTM}(\mathbf{h}_t^{(k)}, \mathbf{h}_{t-1}^{(k+1)}) + \mathbf{h}_t^{(k)} \quad (5.16)$$

In this project, we use LSTM as a recurrent unit, 200 dimensional embedding vector, bi-directional LSTM layer with each direction having 128 units, maximum sentence length of 32. In addition, a dropout unit is added to the LSTM units of keep probability of 0.8, residual connection is applied for deeper networks, and beam search decoding of width 10 is used in inference. During training, the cross entropy is used as the loss function, and scheduled sampling is adopted.

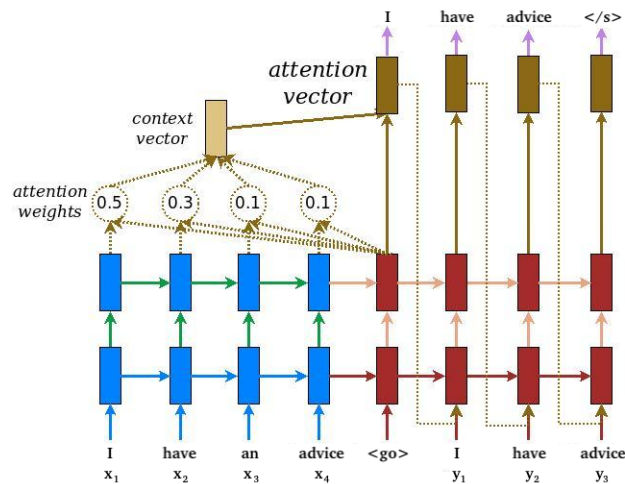


Fig. 5.3 Standard encoder decoder with attention mechanism architecture. Figure adapted from the TensorFlow NMT tutorial: <https://github.com/tensorflow/nmt>.

5.2.2 Implementation

The NMT system is implemented in Python (version 3.6) with TensorFlow³ (version 1.5.0) as they are the most widely used open-source language and library for developing and training machine learning models. TensorFlow library provides the basic building blocks such as RNN units and attention mechanism cells. The source code developed for this project is made available on GitHub⁴ which could also be used for any related machine translation task.

³<https://www.tensorflow.org/>

⁴<https://github.com/potsawee/seq2seq>

5.3 Evaluation Metrics

M² scorer

The M² scorer [40] evaluates system performance by how well its hypothesis or its edits match the gold-standard edits. It computes the sequence of phrase-level edits between a source sentence and a system’s candidate that achieves the highest overlap with the gold-standard annotation. Similar to the computation of precision and recall for error detection in Section 4.2, we compute:

$$P = \frac{\sum_{i=1}^n |\mathbf{g}_i \cap \mathbf{e}_i|}{\sum_{i=1}^n |\mathbf{g}_i|} \quad (5.17)$$

$$R = \frac{\sum_{i=1}^n |\mathbf{g}_i \cap \mathbf{e}_i|}{\sum_{i=1}^n |\mathbf{e}_i|} \quad (5.18)$$

where \mathbf{g}_i is the set of gold-standard edits for sentence i , and \mathbf{e}_i is the set of system edits for sentence i . F-score is defined the same as Equation 4.6. The intersection between \mathbf{g}_i and \mathbf{e}_i is defined as

$$\mathbf{g}_i \cap \mathbf{e}_i = \{e \in \mathbf{e}_i | \exists g \in \mathbf{g}_i, \text{match}(g, e)\} \quad (5.19)$$

This project uses the official scorer tool⁵ (version 3.2) of the CoNLL 2014 shared task [13].

5.4 Experiments

5.4.1 LM-based GEC

Three types of language models are used throughout this project: n-gram, RNN, and suRNN. In this chapter, all of the language models are trained on the one-billion corpus. The 4-gram and three variants of recurrent neural network language models are trained:

- RNN1: layers = 64k:300i:300g:64k
- RNN2: layers = 64k:300i:300g:300f:64k
- suRNN: layers = 64k:300i:300g:300f:64k

where i=linear projection layer, g=GRU based recurrent layer, and f=feedforward layer. The vocabulary size is 64k. We use the CUED-RNNLM toolkit⁶ for training RNN language models.

⁵<https://github.com/nusnlp/m2scorer>

⁶<http://mi.eng.cam.ac.uk/projects/cued-rnnlm/>

LM-based GEC without annotated data for training

Firstly, we investigate the LM-based system without using any *annotated* data. We train the RNN1 language model on the one-billion corpus. Note that different types of language models will be compared in the following experiment. This form of the LM-based system achieves $F_{0.5}$ scores of 26.2 on CoNLL and 18.3 on BULATS. Without using more data, we tune the bias term added to the arcs corresponding to original words, resulting in a small gain in $F_{0.5}$ on both evaluation sets, as shown in Figure 5.4.

LM-based GEC with GED

To improve the system, we consider the use of GED for confusion set generation. We use RNN1 language model as in the previous experiment and the GED system trained on the CLC data without fine-tuning described in Chapter 4. The GED system operates at the 0.5 error threshold. By applying GED for confusion set generation, it is found that both precision and recall increase, and the $F_{0.5}$ score improves by approximately 3.3 on CoNLL and 3.2 on BULATS. Then, we tune the bias term for this system. The results in Figure 5.5 show that we can increase the $F_{0.5}$ score by approximately 0.92 on CoNLL, whereas the $F_{0.5}$ score increases by less than 1.0 on BULATS.

LM-based GEC	CoNLL			BULATS		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
baseline	34.4	13.4	26.2	28.0	7.7	18.3
+ bias	35.2	13.3	26.5	28.2	7.7	18.4
+ GED	40.4	14.2	29.5	31.4	9.5	21.5
+ GED + bias	43.5	13.8	30.4	31.7	9.5	21.6

Table 5.1 LM-based (RNN1) GEC performance on CoNLL and BULATS

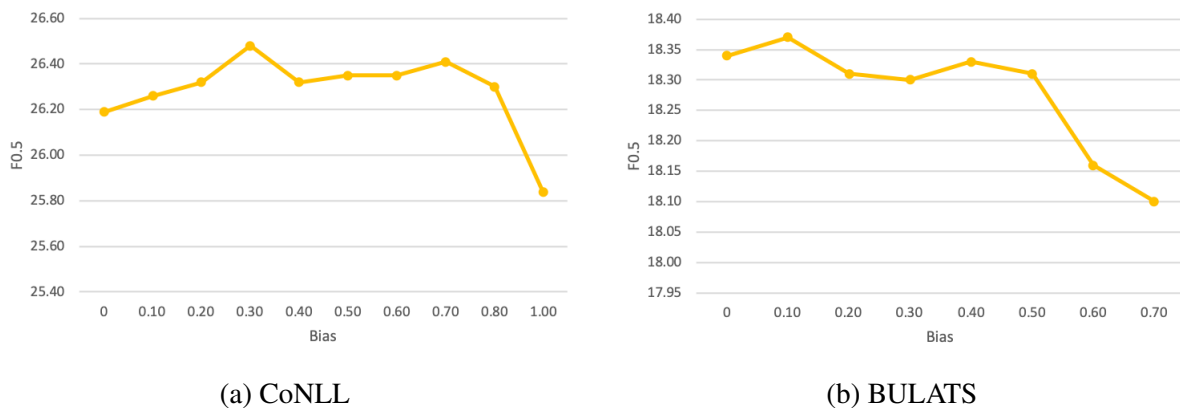


Fig. 5.4 $F_{0.5}$ of LM-based (RNN1) system at different values of bias term

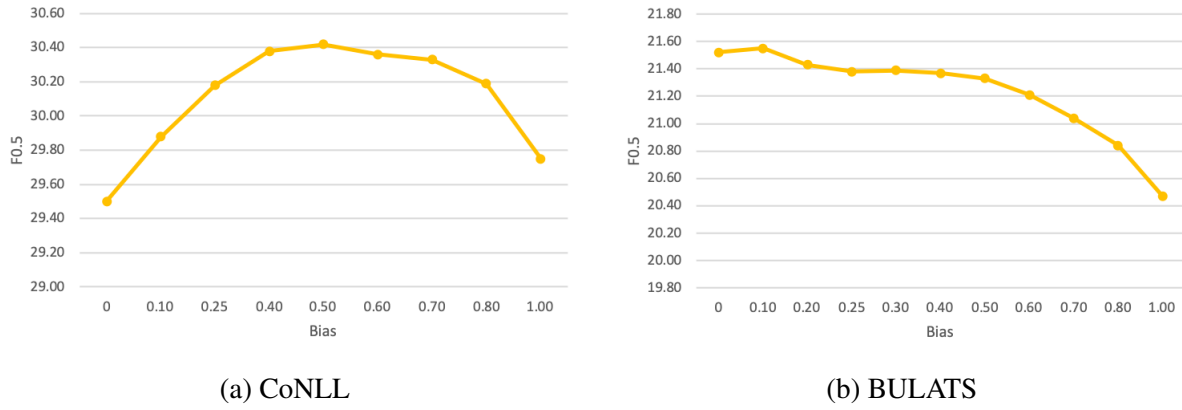


Fig. 5.5 F_{0.5} of LM-based (RNN1) system with GED at different values of bias term

RNNLM-based system at various operating points of GED

Since confusion sets are generated for words that are detected as errors by the GED system, it introduces a trade-off between precision and recall of the GEC system. This is because as GED’s recall increases, more words are inflected into their candidates, resulting in larger lattices; hence, higher GEC’s recall. However, as the lattices become larger, the language model makes more mistakes in selecting the correct path. The RNN1 language model is used. By varying GED’s threshold, Figure 5.6 verifies the trade-off between precision and recall of GEC. For CoNLL, the operating point of 16.3% recall rate or 0.50 error threshold yields the highest F_{0.5}, and for BULATS, that is 33.1% recall rate or 0.55 error threshold. Thus, when applying the GED system in the subsequent experiments, the 0.50 error threshold will be chosen.

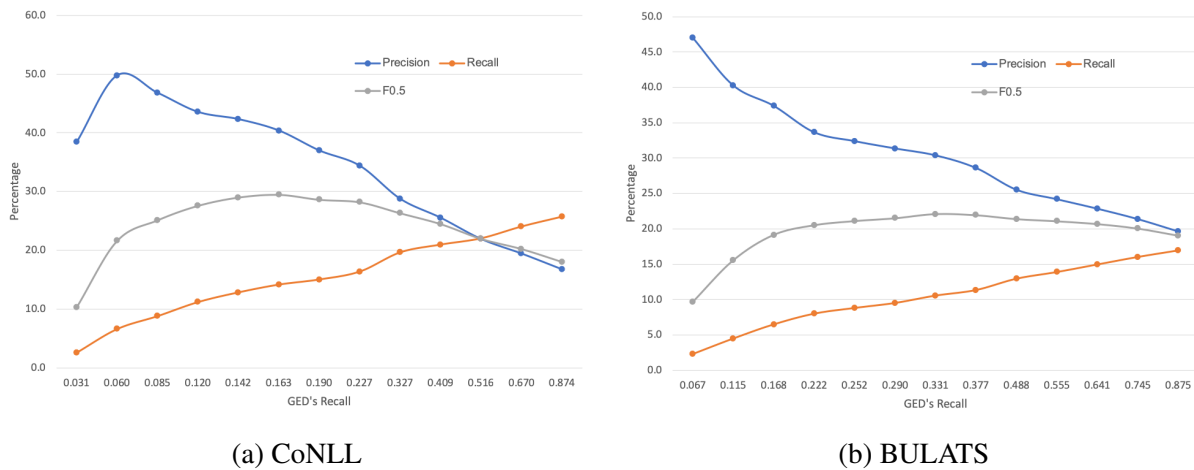


Fig. 5.6 LM-based (RNN1) GEC system at various GED’s operating points.

N-gram and RNN Language Models for GEC

The use of language models is crucial for this approach, so it is worthwhile to compare between different types of language models for lattice rescoring. The LM-based GEC with GED operating at the 0.50 error threshold is used. We train 4-gram, RNN1, RNN2, and suRNN language models

on the one-billion corpus. In Table 5.2, the results show that the 4-gram model has the highest precision score, but the lowest $F_{0.5}$ score among all models. When comparing the recurrent language models, the state-of-art suRNN model has the highest precision, recall, and $F_{0.5}$ score. RNN2, which has an additional layer to RNN1, has a lower performance level than RNN1. In conclusion, we can improve the previous LM-based system by replacing RNN1 with suRNN.

LM	CoNLL			BULATS		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
4-gram	44.7	12.0	28.9	34.5	8.3	21.1
RNN1	40.4	14.2	29.5	31.4	9.5	21.5
RNN2	39.8	14.1	29.2	31.0	9.5	21.4
suRNN	42.4	14.3	30.5	32.0	9.6	21.9

Table 5.2 LM-based GEC (with GED) with different LMs trained on the one-billion corpus

GEC with an idealised GED

Because applying GED to the pipeline yields a gain of around 3.2 in $F_{0.5}$, this experiment aims to determine the maximum gain we could achieve given an idealised GED system. As the labels of words in both CoNLL and BULATS data sets can be determined, it is possible to use this information to expand all the words that are being grammatically incorrect, hence, idealised or perfect GED. It should be noted that the GEC performance is now limited by the coverage of AGID, which will be discussed in the next part, and the language model. As shown in Table 5.3, we could achieve an improvement in $F_{0.5}$ at most around 10 to 15 by improving the GED system. This may not be the best option as a large amount of in-domain error-tagged data would be required to achieve the idealised GED. In [23], the best performing GED system evaluated on BULATS has the $F_{0.5}$ of 55.8, which is still far from being ideal.

LM	CoNLL			BULATS		
	GED	GED*	gain	GED	GED*	gain
RNN1	29.5	39.9	10.4	21.5	36.6	15.1
RNN2	29.2	40.3	11.1	21.4	36.2	14.8
suRNN	30.5	40.8	10.3	21.9	36.7	14.8

Table 5.3 $F_{0.5}$ of LM-based using trained GED (denoted by GED) and with idealised GED (denoted by GED*)

Coverage of AGID

Using the LM-based approach, although GEC’s precision could be as high as 100, GEC’s recall is limited by the AGID, and this could be measured by **coverage**. The coverage is the percentage of the errors that can be corrected by the AGID and the pre-defined sets; hence, this represents the upper bound of the GEC’s recall. In Table 5.4, the coverage of deletion errors is zero because the current LM-based system only derives inflected forms of a word. Insertion errors and multiple-edit errors are grouped into the complex error type. Due to the nature of BULATS that has more insertion errors than CoNLL, the higher coverage of complex errors in BULATS is observed.

Data	Substitution	Deletion	Complex	Overall
CoNLL	63.1	0.0	39.3	51.2
BULATS	79.5	0.0	69.0	56.7

Table 5.4 Coverage of AGID on different data sets

By considering the overall coverage, given an idealised language model, the maximum recall rate that the LM-based approach can achieve is 51.2 on CoNLL, and 56.7 on BULATS. In fact, the actual recall is lower due to word re-ordering in correction.

5.4.2 NMT-based GEC

Training details

In contrast to the LM-based approach, the NMT-based approach requires pairs of source (erroneous) and target (corrected) sentences for training. In this project, we use the CLC.

Bi-LSTM with attention mechanism

We train the baseline NMT system which is the bi-directional encoder LSTM with the attention mechanism. Dropout, beam search decoding, scheduled sampling, and pre-trained embedding techniques are then added to the NMT system. We evaluate the NMT system on CoNLL and BULATS for each modification. Table 5.5 shows the following:

- *Dropout* improves the precision of the system although the recall drops; this suggests that it makes fewer mistakes in correction which is desirable for feedback systems.
- *Beam search decoding* improves both the precision and the recall in both evaluation sets; thus, this decoding method should be used instead of the greedy search decoding.
- *Scheduled sampling* appears to add a small gain in the precision on both sets, whereas the recall drops.

- *Pre-trained embedding* could be an unnecessary feature to be added since both precision and recall decrease on CoNLL, and it only increases precision marginally on BULATS.

NMT model	CoNLL			BULATS		
	P	R	F _{0.5}	P	R	F _{0.5}
baseline NMT	32.5	25.3	30.8	38.6	21.4	33.2
+ dropout	34.9	23.7	31.9	39.8	19.5	32.9
+ beam search decoding	35.4	24.1	32.4	41.3	20.2	34.2
+ scheduled sampling	36.2	23.4	32.6	41.4	19.7	33.9
+ pre-trained embedding	35.0	21.7	31.2	41.8	18.4	33.3

Table 5.5 GEC performance of NMT systems

Training Deeper Models

In this experiment, we are interested in whether having more layers will improve the system. The NMT system with scheduled sampling is extended to 4-layer and 6-layer models. Note that, the number of layers is even because a bi-directional LSTM encoder is used. The first two rows in Table 5.6 show that vanilla deeper models have lower performance than the 2-layer model. We then apply residual connection and decaying learning rate techniques to these deeper models, and the results show that these techniques improve the performance of the deeper models; however, they are only comparable or worse than the 2-layer model.

In [41], it is shown that deeper NMT models give better results on machine translation tasks. In contrast, we find the opposite result for GEC on the CoNLL and BULATS evaluation sets. Although deeper models are more capable of capturing more error patterns and longer dependencies, they are more prone to the over-fitting problem. Also, for GEC, many grammatical errors are identified by a few words nearby rather than longer sequences. As a result, the deeper models do not outperform the 2-layer model in this experiment.

NMT model	CoNLL			BULATS		
	P	R	F _{0.5}	P	R	F _{0.5}
4-layer	33.4	22.9	30.6	40.6	20.1	33.7
6-layer	32.2	18.8	28.2	37.8	15.4	29.3
4-layer + residual	33.5	24.4	31.1	40.4	19.9	33.5
6-layer + residual	32.2	22.8	29.8	39.0	18.4	31.9
4-layer + residual + decaying α	36.0	23.0	32.3	42.1	19.4	34.1

Table 5.6 Training Deeper Models

Transfer Learning

In [23], the GED system is fine-tuned to the BULATS data by performing 10-fold cross-validation, which increases the $F_{0.5}$ score by 11.7. This result motivates us to apply transfer learning to our NMT systems. The BULATS data set is shuffled and split into 5 distinct subsets. A 5-fold cross-validation experiment was carried out where 4 subsets are used for fine-tuning the NMT system trained on the CLC, and the other subset is held-out for evaluation. The overall result is obtained by combining all held-out blocks together before scoring against the reference. This process is also carried out on the CoNLL data set, where the corrected (target) sentences are chosen from the first annotator.

As shown in Table 5.7, the transfer learning method yields **4.8** and **7.2** improvements in $F_{0.5}$ on the two evaluation sets. The larger improvement on the BULATS data compared to the CoNLL data could be because the difference between the CLC (written-text) and BULATS (prompt responses) is more significant than that between the CLC (written-text) and CoNLL (written-text).

NMT system	CoNLL			BULATS		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
baseline	36.2	23.4	32.6	41.4	19.7	33.9
set 1	47.5	22.5	38.8	47.6	24.9	40.3
set 2	43.9	22.7	37.0	49.3	26.8	42.2
set 3	46.8	21.3	37.7	48.0	23.6	39.7
set 4	43.9	23.4	37.3	49.7	25.8	41.9
set 5	41.8	23.4	36.1	49.9	24.7	41.5
overall	44.7	22.6	37.4	48.9	25.2	41.1

Table 5.7 NMT system with transfer learning on CoNLL and BULATS data sets

5.5 Comparison between the two approaches

We have investigated two approaches for GEC on two data sets: written-text (processed) CoNLL and the manual transcription of BULATS data. This section provides the main findings from the experiments and compares the two approaches. In summary, for the LM-based approach, it suffers a low recall rate and hence, low $F_{0.5}$ score because it cannot correct deletion errors, and it cannot handle word-reordering. A GED system can be applied in the confusion set generation stage, resulting in a gain of about 3.2 in $F_{0.5}$. Tuning bias for LM-based systems only gives a small gain in $F_{0.5}$, lower than 1.0, due to the trade-off between precision and recall. For the NMT-based system, it consistently has a higher performance level than the LM-based system,

especially on BULATS, at the cost of obtaining parallel corpora for training. Transfer learning can be applied to the NMT system giving a large gain in $F_{0.5}$ on both evaluation sets.

When comparing the two approaches, Figure 5.7 shows that without using annotated data, the LM-based approach could achieve $F_{0.5}$ scores of 26.5 and 18.4 on CoNLL and BULATS respectively. Given data with error tagged for GED, which is more straightforward to obtain than data annotated for GEC, is available, they can be used to increase the performance of the LM-based. The increase found on CoNLL makes the LM-based system merely worse than the NMT-based system by around 2 in $F_{0.5}$; however, on BULATS, the LM-based system remains at a low performance level even with GED. This could be because the LM-based system cannot correct deletion errors and re-order words which occur more frequently in spoken language. If data annotated for GEC is available, the NMT-based approach should be chosen as the NMT system without transfer learning outperforms the LM-based approach using GED and tuning bias on both evaluation sets. Furthermore, if the matched data is available, one could perform transfer learning on the NMT system, which yields the best performing system. Moreover, Section 6.2 will present further improvement by performing GEC system combination.

GEC system	Training data
LM	Unannotated
LM + GED	Unannotated + Error Tagged
NMT	Corrected
NMT + TL	Corrected + Corrected (target domain)

Table 5.8 Summary of data required to train different GEC systems. Note that corrected data refers to parallel texts consisting of the corrected version and the erroneous version. TL denotes transfer learning.

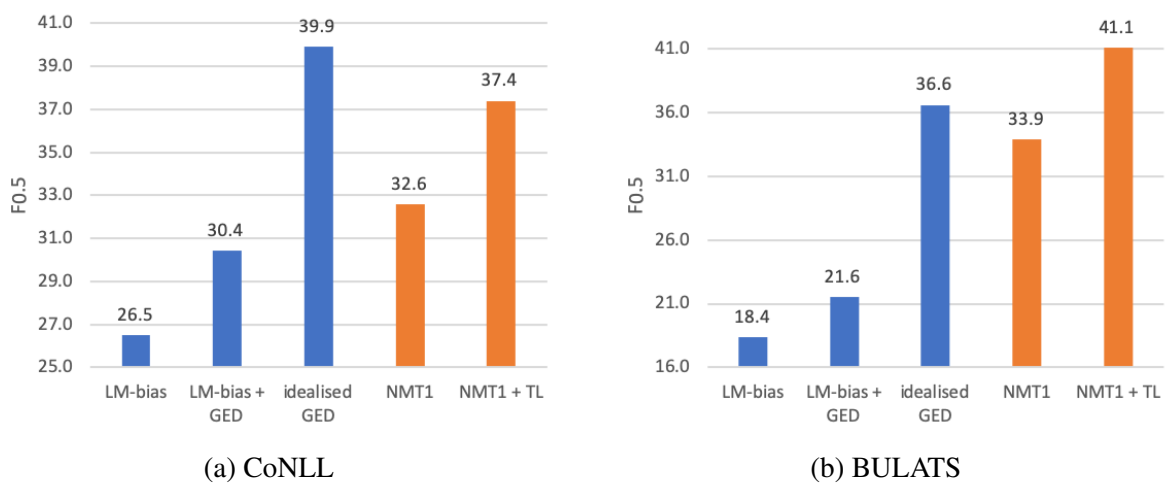


Fig. 5.7 Summary of the GEC systems where LM-based system uses RNN1 language model, NMT1 is the bi-LSTM model using beam search decoding, dropout, and scheduled sampling. TL denotes transfer learning.

Chapter 6

Fluency Score

The fluency score is defined as the log-probability of a sentence computed by a language model:

$$F(w_{1:N}) = \frac{1}{N} \log p(w_{1:N}) = \frac{1}{N} \sum_{i=1}^N \log p(w_i | w_{i-1}, \dots, w_1) \quad (6.1)$$

Provided that the same language model is used, this score can be used as a relative measure of how likely the sentence is. [42] illustrates the use of a language model for evaluating the fluency of a sentence. A sentence’s fluency can be used in both training and inference. During training, less fluent sentences could be generated from grammatical sentences using either n-gram statistics or reverse NMT, and only those with lower fluency scores are kept for further training GEC systems. In the inference stage, the score could be used for selecting the most fluent sentence from a list of candidates generated by multiple GEC systems in combination, and it could be used for re-ordering candidates in the beam search decoding. The probability can be approximated using an n-gram or RNN language model as described in Section 5.1.3. This empirical investigation aims to obtain a language model such that the difference of the scores between *more fluent* and *less fluent* text is as large as possible.

$$\delta(w_{1:N}^{(c)}, w_{1:N}^{(i)}) = F(w_{1:N}^{(c)}) - F(w_{1:N}^{(i)}) \quad (6.2)$$

This chapter includes two examples of the application of the Fluency Score: GEC system combination, and NMT re-ordering.

6.1 The distribution of the Fluency Score on the CLC

The number of sentences in the CLC is 874K, of which 380K (43%) contains no grammatical errors, so in this experiment, 494K sentences with grammatical errors are investigated. The difference is computed as defined in Equation 6.2. Out-of-vocabulary (OOV) words result in unexpectedly low fluency scores; hence, if the number of OOV words in a sentence is different from that of its pair, this sentence pair is rejected, and not included in the calculation. They are

shown in Tables 6.1 and 6.2 as Rejected1 and Rejected2; Rejected1 is when the number of OOV words in an error-corrected sentence is fewer than that of the learner-written sentence in the CLC, and vice versa for Rejected2.

Firstly, a 5-gram language model is trained on each of the corpora to examine which corpus yields the best discrimination. The vocabulary file is built consisting of the words that have more than three occurrences in the one-billion corpus. Table 6.1 shows that the 5-gram model trained on the one-billion corpus gives the highest mean and the highest percentage of positive difference. This result should be expected because the one-billion corpus is by far the largest in size - about 20 times larger than the Fisher corpus.

Training Data	Vocab Size	Rejected1	Rejected2	Mean	Std	+ve %
AMI	18.8K	11.7	6.3	0.117	0.367	52.8
Switchboard	27.5K	11.8	5.3	0.146	0.374	55.4
MGB	51.8K	11.1	2.8	0.179	0.416	59.7
Fisher	68.7K	8.5	2.4	0.210	0.425	63.3
One-Billion	64.0K	9.5	1.7	0.354	0.489	71.5

Table 6.1 Fluency Score results based on 5-gram LM trained on different datasets

Secondly, three variants of language models are trained on the one-billion corpus. The RNNLM is the **suRNN** model described in Section 5.4.1. Table 6.2 shows that the RNNLM is slightly better than the 5-gram LM; both 5-gram and RNN models have the same mean although the RNNLM has lower variance and a higher percentage of positive difference. In practice, as the RNNLM requires more computation, hence, takes longer time, the 5-gram LM trained on the one-billion corpus is selected for subsequent experiments requiring the fluency score.

Language Model	Rejected1	Rejected2	Mean	Std	+ve %
4-gram	9.5	1.7	0.351	0.486	71.4
5-gram	9.5	1.7	0.354	0.489	71.5
RNN	9.5	1.7	0.354	0.478	73.2

Table 6.2 Fluency Score results based on different LMs trained on the one-billion corpus with vocabulary size of 64.0K

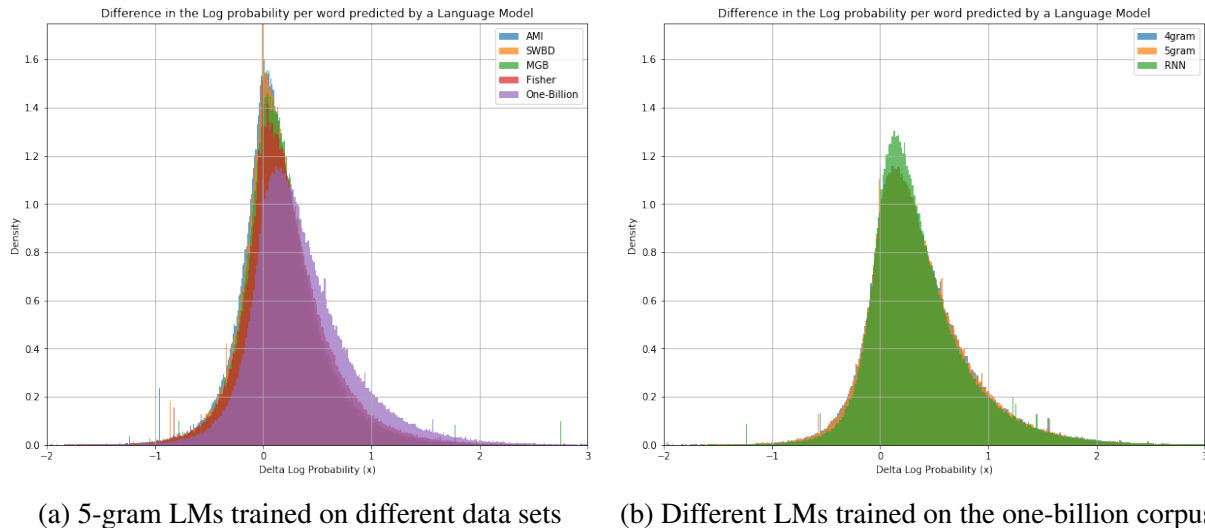


Fig. 6.1 The distribution of the difference of the log-probability

6.2 GEC System Combination

One-best outputs from two systems are combined using the character-level Levenshtein distance [43], which computes the number of edit operations required to change the source sentence into the target sentence. The alignment effectively gives a confusion set, which represents all possible candidates from the two systems. Finally, the most likely sentence from all combinations is selected using the fluency score. For example,

 System1's output:

you prepare chicken with curry and rice

System2's output:

you prepared chicken curry and rice

Alignment:

you	prepare	chicken	with	curry	and	rice
you	prepared	chicken	****	curry	and	rice

Possible Combinations:

you prepared chicken curry and rice
 you prepared chicken with curry and rice
 you prepare chicken curry and rice
 you prepare chicken with curry and rice

6.2.1 Experiments

The one-best outputs of LM-based and NMT-based GEC systems are aligned and combined as described previously. In the implementation, the maximum number of candidates is limited to 1024 to avoid the confusion set being too large. In this experiment, we use the LM-based system as trained in Section 5.4.1. Furthermore, we revisit the training of NMT systems and obtain two models: *baseline-clc* and *large-clc*. The *baseline-clc* system is trained on the same amount of the CLC data as in Section 5.4.1, but the data are further tokenised using the NLTK package in Python. The *large-clc* system is trained on the largest CLC available to this project.

Results

The revised *baseline-clc* NMT system achieves higher $F_{0.5}$ on CoNLL since the processing done is closer to CoNLL, and its performance on BULATS is only slightly higher. The results show that any combination of systems has higher precision than a single system. In Table 6.3, any combined system has higher $F_{0.5}$ than a single system when evaluated on CoNLL. In Table 6.4, due to the significantly low recall rate of the LM-based system on BULATS, the combined systems do not outperform a single NMT system; nevertheless, combining two NMT systems still results in a better system.

System	Variant	P	R	$F_{0.5}$
(1) LM-based	suRNN	42.4	14.3	30.5
(2) NMT-based	baseline-clc	40.1	25.2	35.8
(3) NMT-based	large-clc	42.9	25.8	37.8
(1) and (2)	5gram-onebillion	45.3	24.1	38.5
(1) and (3)	5gram-onebillion	46.4	24.4	39.3
(2) and (3)	5gram-onebillion	43.0	27.3	38.6

Table 6.3 GEC system combination evaluated on CoNLL

System	Variant	P	R	$F_{0.5}$
(1) LM-based	suRNN	32.0	9.6	21.9
(2) NMT-based	baseline-clc	40.3	21.8	34.4
(3) NMT-based	large-clc	42.0	24.5	36.7
(1) and (2)	5gram-onebillion	40.7	18.2	32.6
(1) and (3)	5gram-onebillion	42.3	20.6	35.0
(2) and (3)	5gram-onebillion	42.3	25.7	37.4

Table 6.4 GEC system combination evaluated on BULATS

6.3 NMT-based system Re-ordering

This experiment investigates the potential gain from re-ordering the output of an NMT system. In [44], the output of an SMT-based system for GEC is re-ordered using four types of features, including SMT’s score and LM’s score to train a support vector machine. In this experiment, we experiment the simplest way of re-ordering that is to re-order the 10-best output of a beam-search-decoding NMT system using the LM’s score.

6.3.1 Experiments

We use the `large-clc` NMT system as our baseline, and the output of this system is re-ordered using the 5-gram LM trained on the one-billion corpus. Table 6.5 shows that by re-ordering, although the recall rate increases, the precision rate decreases; overall, the reduction in $F_{0.5}$ suggests re-ordering the output solely on the fluency score does not yield improvement.

System	CoNLL			BULATS		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Baseline	42.9	25.8	37.8	42.0	24.5	36.7
Baseline (Re-ordered)	33.4	33.5	33.4	34.6	29.3	33.4

Table 6.5 Re-ordering the output of NMT evaluated on CoNLL & BULATS

6.4 Summary

This chapter investigated the fluency score that can be used to differentiate more fluent sentences from less fluent ones. Two applications of the score were experimented. The findings suggest the following points:

- The quantity of training data is more *important* than the type of language model.
- Combining the NMT-based system with the LM-based system results in a GEC system that has *higher* precision but *lower* recall. It is found that if the recall of the LM-based system is not significantly lower, the overall $F_{0.5}$ score will be higher.
- Using the fluency score to re-order the output of the NMT system without taking into account the score from beam search decoding should be avoided.

Chapter 7

Data Augmentation

Having a larger amount of training data for both GED and GEC systems generally improves the performance of the systems. However, there is a little amount of spoken language data being annotated for GED and GEC tasks, and this makes it more challenging to develop good systems. On the other hand, the written language corpora which have been annotated for GED and GEC tasks are considerably larger in size, so they could be used to expand the training data for spoken language GED and GEC systems. In [45], a phrase-based SMT system and a pattern extraction based on part-of-speech tags are used to generate artificial data for training GED systems.

In this work, n-gram models and neural machine translation (NMT) models are used to for data augmentation. These two approaches extract error patterns from a source corpus and subsequently propagate the errors onto a target corpus. We investigate two aspects of data augmentation: the statistics of the corrupted corpus, and improving GED systems using the augmented data.

7.1 N-gram Statistics

As explained in section 5.1.3, an n-gram model computes the probability of a word given its previous $(n - 1)$ words. In this section, the n-gram language model is applied to parallel corpora as follows:

$$P(A_i \rightarrow B_i | A_{i-1}, A_{i-2}, \dots, A_{i-(n-1)}) = \frac{C(A_{i-(n-1):i}, B_i)}{C(A_{i-(n-1):i})} \quad (7.1)$$

where $A_{i-(n-1):i}$ is a sequence of length n in the source corpus A, and $(A_{i-(n-1):i}, B_i)$ is the sequence $A_{i-(n-1):i}$ with the final word corresponding to A_i in the target corpus B is B_i . For example,

Unigram Model:

$$P(A_i \rightarrow B_i) = \frac{C(A_i, B_i)}{C(A_i)} \quad (7.2)$$

Bigram Model:

$$P(A_i \rightarrow B_i | A_{i-1}) = \frac{C((A_{i-1}, A_i), B_i)}{C(A_{i-1}, A_i)} \quad (7.3)$$

An example of a parallel corpora where the source data (corpus A) has no error, and the target data (corpus B) contains errors made by learners. A deletion error is denoted by ***.

Source (A)		Target (B)

the	->	the
cat	->	cat
sat	->	sit
on	->	in
the	->	***
mat	->	mat

The form of estimation suggests that although higher-order of n-gram can better capture dependencies between words, counting sequences in a sparse data set results in a poor estimate as there could be only a few occurrences of each sequence.

The distributions of words in two corpora may differ due to multiple factors. For example, one corpus is about business meeting such as AMI, and the other is telephone conversation such as Switchboard or Fisher. In addition, one could be a transcription of speech data, and the other is written text. As a result, some corrupted words appear too frequently, while the other may not appear at all. To illustrate this problem, consider an example of training the unigram model on the CLC, and use the model to corrupt the AMI corpus. Assume that in the CLC,

$$P(\text{stuff} \rightarrow \text{staff}) = 0.1$$

in the AMI corpus,

$$C(\text{stuff}) = 1000, C(\text{staff}) = 10$$

After corrupting, assume that all of the word *staff*'s get mapped *staff*, and no other words get mapped to *staff* apart from *stuff*, the corrupted AMI data will have the following distribution:

$$P(\text{incorrect} | \text{staff}) = \frac{100}{100 + 10} = 0.909$$

Therefore, when a GED system is trained on this corrupted AMI data, it is likely to label the word *staff* as being incorrect with unexpectedly high probability.

One method to mitigate this problem is by balancing distributions for words across the training data (e.g. CLC) and the data to-be-corrupted (e.g. AMI), for the unigram model:

$$P^*(A_i \rightarrow B_i) = \min \left(0.1 \times \frac{C_{\text{corrupted}}(B_i)}{C_{\text{corrupted}}(A_i)}, P(A_i \rightarrow B_i) \right) \quad (7.4)$$

where $C_{\text{corrupted}}(A_i)$ and $C_{\text{corrupted}}(B_i)$ are the counts of words A_i and B_i in the corpus to-be-corrupted. The model in Equation 7.4 will be denoted as the *modified unigram* model. The intuition is that if there are only few counts of word B_i in the corpus to be corrupted, we should limit the transition probability to the ratio $\frac{C_{\text{corrupted}}(A_i)}{C_{\text{corrupted}}(B_i)}$, and the factor 0.1 is introduced because there could be multiple words that are mapped to B_i .

7.2 NMT Statistics

Alternatively, the error statistics can be modelled using a neural network. This is similar to the machine translation problem where a neural network can be used for translating from one to another language. For this task, the source sentence is the error-corrected version, and the target sentence is the erroneous version. This approach is also known as *reverse translation*.

The neural networks used in this experiment are the Bi-LSTM with attention mechanism described in Section 5.2.1. Once the NMT models have been trained, the models can be used to corrupt error-free corpora, i.e. to translate from good English sentences to less grammatical ones. At the inference time, we investigate two decoding methods:

- Sampling: samples from the distribution of the output layer instead of using argmax, and passes the result through an embedding layer to get the next input.
- Beam search: explores the search space of all possible translations by keeping a small set of top candidates, e.g. 10, as the decoding takes place.

7.3 Experiments

7.3.1 Corrupting Native Speech Corpora

In the area of speech processing, there are various native speech corpora available ranging from the AMI corpus of size 2M words to the Fisher corpus of size 35M words. Hence, this experiment aims to investigate the extent to which we can corrupt these corpora using error statistics from the CLC to build a system to work on speech data such as BULATS.

Grammatical Error Statistics

Table 7.1 shows that the BULATS data set, which is speech, has almost twice as many grammatical errors as the CLC, which is written text, with the insertion error rate being about four-fold higher. This illustrates one difference between speech and written text that when learners speak, they are more likely to make mistakes by saying unnecessary or missing words. However, word-choice mistakes as shown by the substitution error percentage are not significantly different.

Corpus	%Error	%Substitution	%Insertion	%Deletion
CLC	10.40	5.72	1.99	2.70
AMI	8.20	4.01	1.52	2.66
Switchboard	8.46	4.15	1.62	2.70
MGB	8.88	4.61	1.55	2.72
Fisher	8.11	3.96	1.61	2.54
BULATS	19.08	6.20	7.94	4.95

Table 7.1 Statistics of the CLC, BULATS, and native-speech corpora after corrupting using the modified unigram statistics.

7.3.2 GED with Data Augmentation

Setting

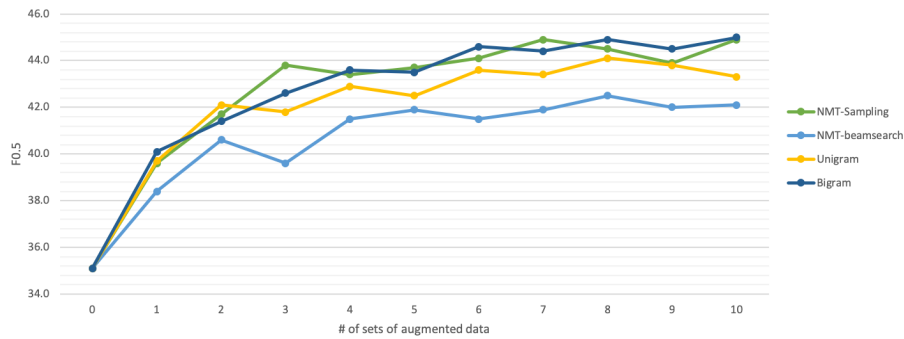
We investigate five models for error generation: unigram, modified unigram, bigram, NMT with sampling decoding, and NMT with beam search decoding. All of the models are trained on the CLC, and the trained models are used for corrupting grammatically correct data. We decide to corrupt: (1) the corrected version of the FCE-public training set which matches the CLC, and (2) the manual transcription of the Switchboard corpus which does not match the CLC, but represents speech data. The GED system with augmented data is evaluated on: (1) the FCE-public test set, (2) the BULATS data set, and (3) the NICT-JLE data set. The baseline GED model is trained on the FCE-public training data set. Then, each of the different versions of the corrupted data is added in addition to the FCE-public training set to train a GED system. We repeat the addition of corrupted data 10 times. Five settings shown in Table 7.2 are investigated.

Results

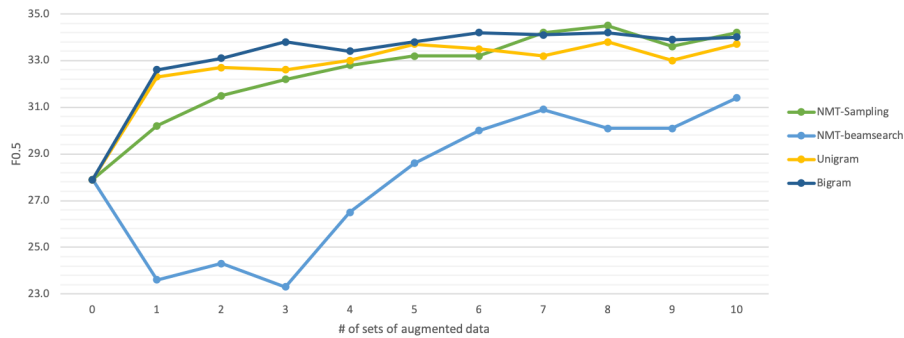
The empirical findings show that the performance of GED increases by using augmented data from any of the corruption models in any experimental setting, as shown in Figure 7.1. Furthermore, the results show that the bigram model is more suitable than both the unigram and the modified unigram model. This could be because the unigram models do not make use of the context of errors, in other words, they only learn the likelihood of each word being incorrect from the CLC and propagate this information onto the target data. For the NMT corruption method, Figure 7.1 shows that in all the settings, the beam search decoding method is always worse than sampling decoding, and NMT with beam search decoding and bigram models are comparable, and they are the most suitable models for error generation. Lastly, when comparing the gain in setting (2) to the gain in setting (3), we can see that augmenting the switchboard data, which are more similar to the BULATS data than the FCE-public data are, gives a higher gain. A similar trend can be seen when comparing setting (4) and (5) which evaluate the system on NICT-JLE.

Setting	Corrupted	Eval	Highest $F_{0.5}$					
			baseline	ugrm	ugrm*	bgrm	samp	beam
(1)	FCE	FCE	35.1	44.1	43.1	45.0	44.9	42.5
(2)	FCE	eval3	27.9	33.8	33.9	34.2	34.5	31.4
(3)	SWBD	eval3	27.9	36.7	37.6	37.4	37.9	33.0
(4)	FCE	NICT	32.7	41.9	41.4	42.2	40.5	37.6
(5)	SWBD	NICT	32.7	43.1	43.1	43.2	41.0	37.4

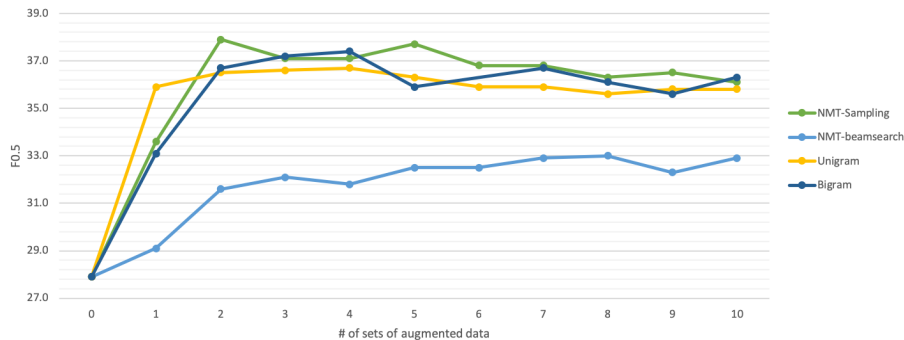
Table 7.2 The highest values of $F_{0.5}$ for different corrupting methods after folding corrupted data 10 times to FCE-public. The statistics is extracted from the CLC for all experiments. Baseline systems were trained on FCE-public without additional data.



(a) Setting 1: $F_{0.5}$ on FCE-public test



(b) Setting 2: $F_{0.5}$ on BULATS



(c) Setting 3: $F_{0.5}$ on BULATS

Fig. 7.1 $F_{0.5}$ for the three experiments as the augmented data increase. Note that the results for settings 4 and 5 follow the same trend as settings 2 and 3.

Chapter 8

Conclusions and Future Work

This project focused on GED and GEC for non-native spoken English. We pre-processed our written-text data such that they became more similar to speech data. The processed written-text data included the FCE-public and CoNLL, which we used as our evaluation sets. Also, our primary spoken-language evaluation set was the manual transcription of the spontaneous spoken responses from the BULATS examination.

Grammatical Error Detection

We treated the problem as a sequence labelling task, and we trained the bi-directional LSTM model on the written-text CLC as our baseline system. We evaluated the baseline system on written and spoken data, and we illustrated the need to improve the system for spoken language purpose. We showed that combining GED systems trained on written text and corrupted native speech data yielded a gain in $F_{0.5}$ on BULATS although this was not as high as the gain achieved by fine-tuning. Future work on GED includes meta-data extraction, which looks at spoken GED from another direction by processing the speech data to be closer to written text data such as removing false starts and repetitions. Besides, a new architecture of the sequence labeller such as pure self-attention mechanism could improve the performance for both written and spoken language data.

Grammatical Error Correction

We developed LM-based GEC and NMT-based GEC systems, and each of them was evaluated on CoNLL and BULATS data sets. The LM-based approach was considered as it does not rely on annotated data for training. Its $F_{0.5}$ score was lower than that of the baseline NMT system on both CoNLL and BULATS. If error-tagged data are available, a GED system can be trained, and by applying the GED system in confusion set generation, the system's performance level went up to almost the same level as that of the NMT system on CoNLL. However, when evaluated on speech data or BULATS, the LM-based system was considerably worse than the NMT system, even when the GED system was used. This is likely because the LM-based system is unable

to correct deletion errors and to re-order word. When comparing language models, we found that the suRNNLM yielded the highest $F_{0.5}$ score on both evaluation sets. The NMT-based approach was shown superior to the LM-based approach, but it requires parallel corpora for training. Moreover, the NMT system can be fine-tuned to the target domain in order to mitigate the problem of domain mismatch, resulting in the best performing system. Future work on GEC includes data augmentation where additional speech training data can be derived from n-gram statistics or reverse NMT. Another interesting way for improvement is the ensemble method.

Fluency Score

Language models can be used for measuring the fluency of a sentence. In this work, it was shown that an essential aspect of obtaining good fluency score is the quantity of training data. The one-billion corpus resulted in a model that best differentiated between correct and incorrect sentences. The type of language models, however, is less critical as there was a small gain obtained from using suRNN instead of 5-gram model. One application that illustrates the use of this score is the GEC system combination, which resulted in a system that outperforms a single GEC system on both evaluation sets. Future work will look at applying this score for filtering reverse NMT for data augmentation.

Data Augmentation

We investigated error generation models based on n-gram and NMT. We found that augmented data from any error generation model increased the performance of GED. When comparing the models, the result showed that the bigram and NMT with sampling decoding models were the most suitable for error generation. Another finding was that augmenting grammatically correct data that are more similar to the target corpus gave a better result. Future work includes applying both the error generation models to improve GEC systems further. Also, the NMT model with beam search may need to be revisited as it performed noticeably worse than the other models.

References

- [1] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [2] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov 1997.
- [3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.
- [4] Julian Richard Medina and Jugal Kalita. Parallel attention mechanisms in neural machine translation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 547–552. IEEE, 2018.
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
- [7] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language_understanding_paper.pdf, 2018.
- [8] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.
- [9] Diane Nicholls. The Cambridge Learner Corpus - error coding and analysis for lexicography and ELT. In *Proc. of the Corpus Linguistics 2003 conference; UCREL technical paper number 16.*, 2003.
- [10] Andrew Caines, Diane Nicholls, and Paula Buttery. Annotating errors and disfluencies in transcriptions of speech. Technical Report UCAM-CL-TR-915, University of Cambridge Computer Laboratory, Dec 2017.
- [11] K.M. Knill et al. Impact of asr performance on free speaking language assessment. In *INTERSPEECH*, pages 1641–1645, 2018.
- [12] Emi Izumi, K. Uchimoto, and H. Isahara. The NICT JLE Corpus Exploiting the language learners’ speech database for research and education. *International Journal of The Computer, the Internet and Management*, 12(2):119–125, May 2004.

- [13] T. Briscoe et al. Ng Hwee Tou, Wu Siew Mei. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task)*, 2014.
- [14] Iain McCowan, Jean Carletta, and et al. Kraaij. The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88, page 100, 2005.
- [15] J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: telephone speech corpus for research and development. In *Proceedings ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992.
- [16] Christopher Cieri, David Miller, and Kevin Walker. The fisher corpus: A resource for the next generations of speech-to-text. 01 2004.
- [17] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013.
- [18] Edward Loper and Steven Bird. NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002.
- [19] Marek Rei and Helen Yannakoudakis. Compositional sequence labeling models for error detection in learner writing. *CoRR*, abs/1607.06153, 2016.
- [20] Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. Attending to characters in neural sequence labeling models. *CoRR*, abs/1611.04361, 2016.
- [21] Ryo Nagata and Kazuhide Nakatani. Evaluating performance of grammatical error detection to maximize learning effect. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.
- [22] Marek Rei. Semi-supervised multitask learning for sequence labeling. *CoRR*, abs/1704.07156, 2017.
- [23] KM Knill, MJF Gales, PP Manakul, and AP Caines. Automatic grammatical error detection of non-native spoken learner english. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8127–8131. IEEE, 2019.
- [24] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al. The htk book. 2002.
- [25] Felix Stahlberg, Christopher Bryant, and Bill Byrne. Neural grammatical error correction with finite state transducers. *CoRR*, abs/1903.10625, 2019.
- [26] X. Chen, X. Liu, Y. Wang, M. J. F. Gales, and P. C. Woodland. Efficient training and evaluation of recurrent neural network language models for automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2146–2157, Nov 2016.
- [27] Yichen Yang. *Large Scale Language Models for Speech Recognition Technical Abstract*. PhD thesis, University of Cambridge, 2018.
- [28] Xie Chen, Xunying Liu, Anton Ragni, Yu Wang, and Mark J. F. Gales. Future word contexts in neural network language models. *CoRR*, abs/1708.05592, 2017.

- [29] Mariano Felice, Zheng Yuan, Øistein E Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, 2014.
- [30] Marcin Junczys-Dowmunt and Roman Grundkiewicz. The amu system in the conll-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33, 2014.
- [31] Anoop Kunchukuttan, Sriram Chaudhury, and Pushpak Bhattacharyya. Tuning a grammar correction system for increased precision. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 60–64, 01 2014.
- [32] Yiming Wang, Longyue Wang, Xiaodong Zeng, Derek F Wong, Lidia S Chao, and Yi Lu. Factored statistical machine translation for grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 83–90, 2014.
- [33] Yonghui Wu, Mike Schuster, and et al. Zhifeng Chen. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.
- [34] Zheng Yuan. *Grammatical error correction in non-native English*. PhD thesis, University of Cambridge, 2017.
- [35] Shamil Chollampatt and Hwee Tou Ng. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [36] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. Approaching neural grammatical error correction as a low-resource machine translation task. *CoRR*, abs/1804.05940, 2018.
- [37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [38] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099, 2015.
- [39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [40] Daniel Dahlmeier and Hwee Tou Ng. Better evaluation for grammatical error correction. In *NAACL*, 2012.
- [41] Antonio Valerio Miceli Barone, Jindrich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. Deep architectures for neural machine translation. *CoRR*, abs/1707.07631, 2017.
- [42] Tao Ge, Furu Wei, and Ming Zhou. Reaching human-level performance in automatic grammatical error correction: An empirical study. *CoRR*, abs/1807.01270, 2018.
- [43] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

- [44] Zheng Yuan, Ted Briscoe, and Mariano Felice. Candidate re-ranking for SMT-based grammatical error correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 256–266, June 2016.
- [45] Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. Artificial error generation with machine translation and syntactic patterns. *arXiv preprint arXiv:1707.05236*, 2017.

Appendix A

Risk Assessment

This project is entirely computer-based, so the risk was mainly due to computer screen use, and the condition of working space such as the position of the screen, table, or chair. The risk assessment evaluated at the beginning of this project reflected these issues well, and there was no further risk that I encountered. If I were to start this project again, I would still assess risk as I did.