ROBUST TEXT CORRECTION

FOR GRAMMAR AND FLUENCY

by

Keisuke Sakaguchi

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2018

Abstract

Grammar is one of the most important properties of natural language. It is a set of structural (i.e., syntactic and morphological) rules that are shared among native speakers in order to engage smooth communication. Automated grammatical error correction (GEC) is a natural language processing (NLP) application, which aims to correct grammatical errors in a given source sentence by computational models.

Since the data-driven statistical methods began in 1990s and early 2000s, the GEC community has worked on establishing a common framework for its evaluation (i.e., dataset and metric for benchmarking) in order to compare GEC models' performance quantitatively. A series of shared tasks since early 2010s is a good example of this.

In the first half of this thesis, I propose character-level and token-level error correction algorithms. For the character-level error correction, I introduce a semi-character recurrent neural network, which is motivated by a finding in psycholinguistics, called the *Cmabrigde Uinervtisy* (Cambridge University) effect or typoglycemia. For word-level error correction, I propose an error-repair dependency parsing algorithm for ungrammatical texts. The algorithm can parse sentences and correct grammatical errors simultaneously.

ABSTRACT

However, it is important to note that grammatical errors are not usually limited to morphological or syntactic errors. For example, collocational errors such as *quick/fast food and *fast/quick meal are not fully explained by only syntactic rules. This is another important property of natural language, called *fluency* (or *acceptability*). Fluency is a level of mastery that goes beyond knowledge of how to follow the rules, and includes knowing when they can be broken or flouted. In fact, the GEC community has also extended the scope of error types from closed class errors (e.g., noun numbers, verb forms) to the fluency-oriented errors.

The second half of this thesis investigates GEC while considering fluency as well as grammaticality. When it comes to "whole-sentence" correction, by extending the scope of errors considering fluency as well as grammaticality, the GEC community has overlooked the reliability and validity of the task scheme (i.e., evaluation metric and dataset for benchmarking). Thus, I reassess the goals of GEC as a "whole-sentence" rewriting task while considering fluency. Following the fluency-oriented GEC framework, I introduce a new benchmark corpus that is more diverse in various aspects such as proficiency, topics, and learners' native languages.

Based on the fluency-oriented metric and dataset, I propose a new "whole-sentence" error correction model with neural reinforcement learning. Unlike conventional maximum likelihood estimation (MLE), the model directly optimizes toward an objective that considers a sentence-level, task-specific evaluation metric. I demonstrate that the proposed model outperforms MLE in human and automated evaluation metrics.

ABSTRACT

Finally, I conclude the thesis and outline ideas and suggestions for future GEC research.

Primary Readers and Advisors: Benjamin Van Durme and Matt Post

Secondary Reader: Philipp Koehn

Acknowledgments

First and foremost, Benjamin Van Durme and Matt Post were great advisors, and I am truly grateful to both of them for taking me as a student. When I faced many challenges during my Ph.D. study, they were always supportive and helped me to get over the difficulties. Ben's unwavering support allowed me to pursue my research interest with a lot of freedom. His wide and deep knowledge of computer science, linguistics, cognitive science, and the philosophy of language was always inspiring to me. Matt taught me how to write a solid research paper, write (and maintain) beautiful code, create beautiful slides, and make good presentations. He always helped me to untangle complicated issues and make them into a simple but strong core research question from a wide perspective. Thank you, Ben and Matt. I am very proud to be one of your students. I can't thank you enough for everything you have done for me.

I want to also thank Philipp Koehn for being a member of my thesis committee. I learned a lot of ideas from machine translation and applied them to my task. Also, I really enjoyed attending his machine translation reading group.

I would like to say thanks to Jason Eisner and his reading group members for the in-

tellectual and interesting discussions about various topics in natural language processing and machine learning. I have learned a lot about dependency parsing and reinforcement learning in this reading group. This knowledge helped to make my thesis more technically solid.

I also would like to give special thanks to Kevin Duh for always being concerned about me, ever since I was a student at Nara Institute of Science Technology (NAIST). When I applied to CLSP, he wrote me a reference letter, which actually led me here. I am so lucky to have worked with you both in Japan and the US.

At JHU, I have been very fortunate to learn a lot through talking with faculty members at the CLSP, HLTCOE, and the Department of Cognitive Science, specifically Raman Arora, Najim Dehak, Mark Dredze, Craig Harman, Hynek Hermansky, Sanjeev Khudanpur, Tom Lippincott, Dan Povey, Suchi Saria, Shinji Watanabe, David Yarowsky, Tal Linzen, Kyle Rawlins, Paul Smolensky and Akira Omaki. Their lectures, seminars, and discussions have stimulated my intellectual curiosity.

I would like to thank Ben's labmates (blab) and my classmates and colleagues for their support in various ways, from chitchat to deep technical discussions, including Nick Andrews, Adrian Benton, Charley Beller, Annabelle Carrell, Yuan Cao, Tongfei Chen, Ryan Cotterell, Ryan Culkin, Shuoyang Ding, Seth Ebner, Matthew Francis-Landau, Frank Ferraro, Aparajita Haldar, Katie Henry, Nils Holzenberger, Ting Hua, Jonathan Jones, Huda Khayrallah, Rebecca Knowles, Gaurav Kumar, Chu-Chen Lin, Vimal Manohar, Rebecca Marvin, Chandler May, Arya McCarthy, Hongyuan Mei, Poorya Mianjy, Sebastian Mielke,

ACKNOWLEDGMENTS

Courtney Napoles, Jason Naradowsky, Nanyun Violet Peng, Adam Poliak, Pushpendre Rastogi, Dee Ann Reisinger, Adi Renduchintala, Rachel Rudinger, Rashmi Sankepally, Peter Schulam, Pamela Shapiro, David Snyder, Adam Teichert, Tim Vieira, Dingquan Wang, Felicity Wang, Yiming Wang, Aaron Steven White, Travis Wolfe, Zack Wood-Doughty, Winston Wu, Patric Xia, Hainan Xu, Xuchen Yao, Mahsa Yarmohammadi, Sheng Zhang, Xiaohui Zhang, and many others. Primarily, it was very fortunate for me to work with Courtney Napoles in several projects. Her detailed analyses of language learners' writing are always impressive. Thank you, Courtney.

Ruth Scally, Yamese Diggs, Carl Pupa, and Zack Burwell helped a lot with administrative work. Their help made me more efficient and reduced a huge amount of my stress. Thank you so much.

Off-campus, I am also lucky to work with Joel Tetreault on various aspects of grammatical error correction, with Nitin Madnani and Mike Heilman at ETS as an intern, and with Mary Swift and Bill Murdock at IBM as an intern. Thank you for giving me such great opportunities.

I would like to thank my previous supervisors, Yuji Matsumoto and Mamoru Komachi, at Nara Institute of Science and Technology. I am very lucky to have started my career in natural language processing in their lab.

I would also like to acknowledge that my Ph.D. research was generously funded by the Nakajima Foundation.

My life in Baltimore has been supported by a number of wonderful friends. I would

ACKNOWLEDGMENTS

like to thank my friends Jeff and Judi Seal, Peta Richkus, Haichong (Kai) and Airi Zhang, Hirofumi Kawaguchi, Ryutah Kato, Natsuki Arai, Yuri Amano, Yutaka Nagahata, Jason Naradowsky for their warm support in what has been such an exciting and unpredictable life in Baltimore.

I would like to thank my parents Yasuyuki and Etsuko, my brothers Shunsuke and Ryosuke, and my parents-in-law Shuichi and Tamiko Senga for always being supportive and concerned about me. I also thank my two month twin boys, Taiga and Leo, for "encouraging" me to write up this thesis as efficiently as possible. Finally, I am grateful for the love and patience from my wife, Nozomi. Without her support, encouragement, and patience, I could not have completed this thesis.

Dedication

This thesis is dedicated to my wife, Nozomi Sakaguchi.

Contents

Abstract **ii**

Acknowledgments **v**

List of Tables **xv**

List of Figures **xvii**

1 Introduction **1**

1.1 Motivation 1

1.2 Brief History of Automated Grammatical Error Correction 3

1.3 Grammaticality and Fluency 6

1.4 An Overview of This Thesis 8

2 Background **12**

2.1 Evaluation Metrics 13

2.2 Methods 19

2.3 Datasets 24

2.4 Summary 29

3 Character-level Error Correction: Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network 31

3.1 Introduction 32

3.2 Raeding Wrods with Jumbled Lettres 34

3.3 Semi-Character Recurrent Neural Network 37

3.4 Character-based Neural Network 40

3.5 Experiments 41

3.5.1 Spelling correction results 42

3.5.2 Corroboration with psycholinguistic experiments 48

3.6 Summary 50

4 Token-level Error Correction: Error-repair Dependency Parsing for Ungrammatical Texts 52

4.1 Introduction 53

4.2 Model 55

4.2.1 Non-directional Easy-first Parsing 55

4.2.2 Error-repair variant of EF 56

4.3 Experiment 59

4.3.1 Data and Evaluation 59

4.3.2 Results 63

4.4 Conclusions 65

5 Reassessing the Goals of Whole Sentence Error Correction 68

5.1 Introduction 69

5.2 Current issues in GEC 72

5.2.1 Annotation methodologies 72

5.2.2 Evaluation practices 75

5.3 Creating a new, *fluent* GEC corpus 77

5.3.1 Data collection 79

5.3.2 Human evaluation 80

5.4 What is the Best Annotation–Evaluation Combination? 83

5.4.1 Experiments 84

5.4.2 Results 85

5.5 GEC System Evaluation by Non-experts 89

5.5.1 Experiments 90

5.5.2 Results 91

5.6 Conclusion 92

6 A Fluency Corpus and Benchmark for Grammatical Error Correction 95

6.1 Introduction 96

6.2 The JFLEG corpus 97

6.3 Evaluation 98

6.4 Conclusions 101

7 Sentence-level Error Correction: Neural Reinforcement Learning for sentence level GEC 103

7.1 Introduction 104

7.2 Model and Optimization 106

7.2.1 Maximum Likelihood Estimation 108

7.2.2 Neural Reinforcement Learning 109

7.2.3 Reward in Grammatical Error Correction 111

7.2.4 Minimum Risk Training and Policy Gradient in Reinforcement Learning 111

7.3 Experiments 115

7.3.1 Data 115

7.3.2 Hyperparameters 116

7.3.3 Baselines 116

7.3.4 Evaluation 118

7.3.5 Results 119

7.3.6 Analysis 121

7.4 Summary 121

8 Conclusions and Future Directions 122

A Efficient Elicitation of Annotations for Manual System Evaluation **129**

A.1 Introduction 130

A.2 Models 133

A.2.1 Expected Wins 134

A.2.2 The Hopkins and May (2013) model 136

A.2.3 TrueSkill 139

A.2.4 Data selection with TrueSkill 144

A.3 Experimental setup 145

A.3.1 Datasets 145

A.3.2 Perplexity 146

A.3.3 Accuracy 147

A.3.4 Parameter Tuning 148

A.4 Reduced Data Collection with Non-uniform Match Selection 149

A.5 Clustering 151

A.6 Conclusion 152

Vita **182**

List of Tables

1.1 Examples of the six base error types in the Helping Our Own (HOO) shared task (adapted and modified from Dale, Anisimoff, and Narroway (2012)). . 4
1.2 Four quadrant of grammaticality and fluency. 7
1.3 Four quadrant of grammaticality and fluency from the NUCLE language learner corpus. 7

2.1 Metric scores of three artificially contrived systems, input source sentences, and top three system outputs. 17
2.2 GEC corpora and the basic information 24
2.3 GEC corpora available for free and several desired properties. 25
2.4 Proportion of errors in Cambridge Learner Corpus 27
2.5 Proportion of errors in the NUS Corpus of Learner English 28

3.1 Example sentences and results for measures of fixation excerpt from Rayner et al. (2006). 35
3.2 Example spelling correction outputs for the *Cmabrigde Uinervtisy* sentences. 43
3.3 Spelling correction accuracy (%) with different error types on the entire test set. 44
3.4 Spelling correction accuracy (%) with different error types on the subset of test set (50 sentences). 44
3.5 scRNN accuracy (%) on jumbled word recognition with different BPTT parameters. There were no statistically significant differences among them. 46
3.6 scRNN accuracy (%), the standard deviation, and the size of model file (KB) on jumbled word recognition with respect to the number of units of LSTM. 46
3.7 Example sentences and results for spelling correction accuracy by scRNN variants depending on different jumble conditions. 47
3.8 Error analysis of scRNN variants. 47

4.1 Unlabeled dependency accuracy results with the 5x5 models and test sets. . 63

4.2 Grammaticality scores by 1-4 scale regression model (Heilman et al., 2014). 63
4.3 Successful and failure examples by EREF. The edits are represented by [-deletion-] and {+insertion+}. 65

5.1 Examples and counterexamples of technically grammatical and fluent sentences. 77
5.2 An example sentence with expert and non-expert fluency edits. 77
5.3 An example sentence with the original NUCLE correction and fluency and minimal edits written by experts and non-experts. 81
5.4 Human ranking of the new annotations by grammaticality. 82
5.5 System ranking obtained from human ranking (adapted from Grundkiewicz, Junczys-Dowmunt, and Gillian (2015). 85
5.6 Correlation between the human ranking and metric scores over different reference sets. 86
5.7 Inter-annotator agreement of pairwise system judgments within non-experts, experts and between them. We show Cohen's κ and quadratic-weighted κ.[1] 91

6.1 A sentence corrected with just minimal edits compared to fluency edits. . . 97
6.2 JFLEG annotation instructions. 99
6.3 Scores of system outputs. * indicates no significant difference from each other. 100

7.1 GEC corpora for training 114
7.2 Summary of baselines, MLE and NRL models. 117
7.3 Human evaluation and GLEU evaluation of system outputs on the development and test set. 118
7.4 M^2 ($F_{0.5}$) scores on the dev set. 119
7.5 M^2 ($F_{0.5}$) scores on the test set. 119
7.6 Ratio of pairwise (preference) judgments between NRL and MLE. NRL>MLE: NRL correction is preferred over MLE. NRL<MLE: MLE is preferred over NRL. NRL=MLE: NRL and MLE are tied. 120
7.7 Example outputs by MLE and NRL 120

8.1 GEC corpora available for free (for research purposes) and desired properties. 124

A.1 System rankings presented as clusters (WMT13 French-English competition). 131

List of Figures

2.1 An example of the edit lattice created by token-based dynamic programming. 15
2.2 Illustrative example of the neural encoder-decoder model (adapted from (Sutskever, Vinyals, and Le, 2014)). 22

3.1 Schematic Illustration of semi-character recurrent neural network (scRNN). 33
3.2 Example of the masked priming procedure. 34
3.3 Learning curve of training scRNN with different BPTT parameter (on dev set): first 40k iterations (top) and its enlarged view between 20k and 40k iterations (bottom). 45

4.1 Illustrative example of partial derivation under error-repair easy-first non-directional dependency parsing. 54
4.2 Unlabeled dependency accuracy results with the 5x5 models and test sets (higher is better). 62
4.3 Grammaticality scores by 1-4 scale regression model 64

5.1 An ungrammatical sentence that can be corrected in different ways. 72
5.2 Mean GLEU scores with different numbers of fluency references. The red line corresponds to the average GLEU score of the 12 GEC systems and the vertical bars show the maximum and minimum GLEU scores. 87
5.3 Screenshot of the GEC grammaticality judgment task (HIT). 89
5.4 Output of system rankings by experts and non-experts, from best to worst. Dotted lines indicate clusters according to bootstrap resampling ($p \leq 0.05$). 91

7.1 Illustrative example of the neural encoder-decoder model with attention (adapted from Bahdanau, Cho, and Bengio (2014)). 106
7.2 Illustration of the exposure bias. 107

A.1 TrueSkill's v_{win} and the corresponding *loss function* in the Hopkins and May model as a function of the difference t of system means. 141

A.2 TrueSkills v_{tie} and the corresponding *loss function* in the Hopkins and May model as a function of the difference t of system means ($\epsilon = 0.5, c = 0.3$, and $d = 0.5$). 142
A.3 Heat map for the ratio of pairwise judgments across the full cross-product of systems in the WMT13 French-English translation task. 149
A.4 Heat map for the ratio of pairwise judgments across the full cross-product of systems used in the *first* 20% of TrueSkill model. 150
A.5 Heat map for the ratio of pairwise judgments across the full cross-product of systems used in the *last* 20% of TrueSkill model. 150
A.6 The number of clusters according to the increase of training data for WMT13 French-English (13 systems in total). 153
A.7 The result of clustering by TrueSkill model with 1K training data from WMT13 French-English. 154
A.8 The result of clustering by TrueSkill model with 25K training data. 155

Chapter 1

Introduction

1.1 Motivation

Natural language is characterized by its grammar: a set of structural (i.e., syntactic) rules for composing words, phrases, clauses, and sentences. Grammar enables native speakers to communicate with the language smoothly. In English, as shown in the following simple example, the subject-verb-object (SVO) word order rule tells which noun phrase (person) is the subject or the object for the hitting event.

(1) Kim hit Bob.

(2) Bob hit Kim.

Grammar is shared among native speakers to make smooth communication, and to put it the other way around, sentences that contain grammatical errors break the smooth commu-

nication among speakers.

English is the most widely used language, and there are 400 million native speakers in the United States, Britain, and the Commonwealth. However, there are a billion non-native English learners in the world (Guo and Beckett, 2007), and that number is expected to grow at a fast pace. For non-native speakers of English, learning all the English grammar is a huge obstacle, and the resulting grammatical errors often prevent smooth communication with native and non-native speakers in English. Therefore, correcting grammatical errors is important to bridge the gap between native and non-native speakers of English.[1]

To correct grammatical errors in writing, we may often turn to professional editors who are the experts of the English language, but this method is sometimes inefficient with respect to time, and the cost is not always reasonable.[2] Automated grammatical error correction (GEC hereafter) is a computational method that automatically detects grammatical errors and corrects them.[3] Once we train a GEC model, a number of users can benefit from it with little cost and time.

[1] Of course, grammatical error correction is important not only for non-native speakers but also native speakers of English. For example, in some high-stakes environments, such as business, medical, and legal domains, grammatical errors may critically affect the stakeholders. In fact, many professional proofreading services exist.

[2] Again, it often depends on the document's importance. For high-stake domains, people would likely rely on professional editors for the accuracy of edits.

[3] In this thesis, I focus on GEC for English learners' writing as the main scope.

1.2 Brief History of Automated Grammatical Error Correction

GEC is one of the applications in natural language processing (NLP hereafter), and it becomes increasingly important as the population of non-native speakers grows more rapidly. GEC originated in the 1980s, when simple string matching and/or rule-based syntactic analysis were used.[4] In the 1990s and early 2000s, data-driven statistical methods began to overtake the rule-based methods. However, the community did not fully benefit from the paradigm shift because (1) language learner corpora were scarce and corpora is necessary for training models, and (2) the community did not have a common framework (i.e., dataset and metric for benchmarking) for its evaluation, unlike other NLP areas, such as syntactic parsing with the Penn TreeBank (Marcus et al., 1994).

The GEC community began to address these issues in the 2010s. One of the greatest impacts on the community was "Helping Our Own" (HOO), a GEC shared task (Dale and Kilgarriff, 2011; Dale, Anisimoff, and Narroway, 2012) in which participants used the same dataset to train and evaluate their models. The shared tasks allowed us to compare GEC models' performance explicitly. In other words, the community can quantify the progress of developing GEC systems. After completion of the HOO shared tasks, the GEC community continued to run another series of shared tasks in CoNLL (The SIGNLL Conference on Computational Natural Language Learning) in 2013 and 2014 (Ng et al.,

[4] I do not go into extensive detail about early days of GEC research. For more details, I encourage readers to refer to Leacock et al. (2014).

Error Type	Source	Correction
Substitution	I could only travel *on* July	I could only travel *in* July
Deletion	I am looking forward your reply	I am looking forward *to* your reply
Insertion	I have booked a flight *to* home	I have booked a flight home
Substitution	The hotel is located on *a* seaside	The hotel is located on *the* seaside
Deletion	I will give you all information	I will give you all *the* information
Insertion	One of my hobbies is *the* photography	One of my hobbies is photography

Table 1.1: Examples of the six base error types in Helping Out Own (HOO) shared task (adapted and modified from Dale, Anisimoff, and Narroway (2012)). The first three rows contain preposition errors, and the last three rows contain determiner errors.

2013; Ng et al., 2014).

Each of the shared tasks has extended the scope of error types. In the HOO shared tasks, the targets of grammatical errors were mainly prepositions and determiners (Table 1.1), and the CoNLL 2013 added other closed-class error types: noun number, verb form, and subject-verb agreement.[5] These error types are very common in language learners' writings and are relatively easy to define as a classification task. As systems improved and more advanced methods were applied to the task, the definition evolved to include *whole-sentence correction*, or correcting all errors in the CoNLL 2014 shared task (Ng et al., 2014), including word tone errors (e.g., formal or informal), collocation/idiom errors, and so on.

It is important to note that these new error types do not necessarily violate syntactic rules, i.e., grammar. Let's take a look at the following sentences.

(1) *From this scope, social media have shorten our distance.

[5]Technically, deletion errors (e.g., I am looking forward []/to your reply.) are not as simple as the other two types (i.e., substitution and insertion errors) because detecting deletion errors includes searching positions (or indices) in the sentence that potentially have the errors (e.g., the head of a noun phrase).

(2) *From this scope, social media *has* shorten our distance.

(3) From this scope, social media *has shortened* our distance.

(4) From this *perspective*, social media has *shortened* our distance.

(5) From this *perspective*, social media has *shortened the distance between us*.

The first sentence is ungrammatical because of the violation of subject-verb number agreement, and it contains the wrong present perfect form (have + past participle) in English grammar. In the second sentence, the agreement error has been corrected, but it is still ungrammatical because of the present perfect form error. At this point, we can count and compare the number of grammatical errors (i.e., grammaticality) between the two sentences. Before the whole sentence correction in the CoNLL 2014 shared task, the community paid attention to this perspective. Specifically, the systems are expected to detect and correct as many *grammatical* errors as possible. In this sense, the third sentence is perfectly grammatical.

In the fourth sentence, the writer has further edited the word "scope" to "perspective", which makes the sentence sound better to native English speakers. At this point, however, we do not have a way to quantify and compare the differences between sentences (3) or (4) just by grammaticality, i.e., counting the number of violations of grammatical rules. In other words, both sentences are perfectly grammatical, but native speakers still discriminate among them according to their naturalness. We can still make the fourth sentence more natural by changing the phrase "our distance" to "the distance between us" as in the fifth

sentence. This naturalness cannot be explained only by grammaticality, and of course, this fact becomes critical when it comes to evaluating GEC systems in the whole-sentence correction scheme, but the community has overlooked this aspect in the CoNLL 2014 shared task and the following work with the same benchmarking scheme. Therefore, we need another explicit measurement or metric to capture the differences among the five sentences above, and it is what we call *fluency*.

1.3 Grammaticality and Fluency

As we have seen in the previous section, we want a measurement that can capture a sentence's naturalness to native speakers. In linguistics, it is called *fluency* (or *acceptability*).[6] The difference between grammaticality and fluency is demonstrated in Table1.2. The four quadrants show the distinction between grammaticality and fluency for each combination. The first example (a) shows a famous sentence by Chomsky (Chomsky, 1957), which is grammatical but not fluent. The second example (b) is grammatical and fluent, and the third sentence (c) is neither grammatical nor fluent. Finally, the last example (d) is fluent but not grammatical, as we often find in colloquial sentences, speech transcription, etc. Table 1.3 shows corresponding examples for each quadrant in the real GEC context that is extracted from the NUCLE language learner corpus (Dahlmeier, Ng, and Wu, 2013).

It is important to note that these two concepts are not completely orthogonal and independent. Generally, grammatical sentences tend to be more fluent and vice versa. Accord-

[6] Although both terms are often used, I use "fluency" throughout this thesis.

	Not Fluent	Fluent
Grammatical	(a) Colorless green ideas sleep furiously.	(b) Harmless young children sleep quietly.
Not Grammatical	(c) Color green idea furious sleep.	(d) He wouldn't know this world in which we live in.

Table 1.2: Four quadrant of grammaticality and fluency.

	Not Fluent	Fluent
Grammatical	(a) Firstly , someone having any kind of disease belongs to his or her privacy.	(b) In addition, it is impractical to make such a law.
Not Grammatical	(c) It is unfair to release a law only point to the genetic disorder.	(d) I don't like this book, it's really boring.

Table 1.3: Four quadrant of grammaticality and fluency from the NUCLE language learner corpus.

ing to Lyons (1968), grammaticality is defined as a part of fluency that can be explained by the rules, and fluency is a more primitive notion than grammaticality and more speaker-oriented than linguist-oriented.

As we have seen in the previous section, the GEC community has apparently changed the scope of error types from linguistic aspects (e.g., determiner errors, preposition errors, etc.) to natural daily usages (e.g., formal/informal errors, word choice errors, etc.) Despite the pivot, however, the community has kept using the same evaluation scheme based only on grammaticality. Specifically, the community has overlooked the validity and reliability of evaluation metrics and benchmarking datasets with respect to *fluency*.

The main goal of this thesis is to address the concern about "whole-sentence" correction with *fluency* and to improve the performance of GEC based on the reassessed evaluation framework. Of course, we are also interested in conventional character or token-level

grammatical error correction, as we will see in the following chapters, where fluency is not directly taken into account.[7] These traditional error correction tasks will give us a brief overview and important technical challenges in error correction and noisy-text processing in general. In fact, in the real world, we are surrounded by and live with a huge amount of noisy texts, such as OCR (optical character recognition), texts in social media, and speech transcriptions. These texts include various kinds of errors such as typos, neologisms, abbreviations, reparanda and interregna. In this respect, GEC is also regarded as one of the noisy-text processing tasks in NLP. Therefore, taken together, I will discuss more background on the issues in "whole-sentence" correction in the next chapter, followed by the character and token-level correction, and then I will come back to sentence-level correction. A more detailed outline of this thesis is described in the next section.

1.4 An Overview of This Thesis

As briefly mentioned in the previous section, the main goal of this thesis is to reassess "whole-sentence" error correction by considering *fluency* in the evaluation metric and benchmarking dataset and to build fluency-oriented GEC models according to the reassessment. In addition, this thesis proposes character and token-level (conventional) grammatical error correction models to gain a high-level perspective and challenges in noisy text processing in general. This thesis's detailed overview and contributions are summarized as follows:

[7] It depends on the scope of the task. Spelling and grammar checking are still important tasks.

Chapter 2 introduces relevant work in whole-sentence GEC. In particular, the chapter summarizes the field of GEC in recent years regarding evaluation metrics, datasets, and methods. The chapter also briefly presents important issues that prior work has overlooked.

Chapter 3 focuses on character-level errors. Primarily, this chapter proposes a robust word recognition model for character-level errors. In psycholinguistic literature, it has been shown that humans have a robust word processing mechanism, where some jumbled words (e.g., Cmabrigde / Cambridge) are recognized with little cost, but computational models for word recognition (e.g., spelling checkers) perform poorly on data with such noise, which is called the *Cmabrigde Uinervtisy* (*Cambridge University*) effect. Inspired by the findings from the *Cmabrigde Uinervtisy* effect, I propose a word recognition model based on a semi-character recurrent neural network (scRNN). In the experiments, I demonstrate that the scRNN performs significantly more robustly in spelling correction (i.e., word recognition) than existing spelling checkers and character-based convolutional neural networks. Furthermore, I demonstrate that the model is cognitively plausible by replicating a psycholinguistics experiment on human reading difficulty with the model.

Chapter 4 moves on to token-level error correction. In this chapter, I propose a parsing algorithm that can correct token-level grammatical errors and parse dependency structures jointly. The algorithm is an extension of Goldberg and Elhadad (2010)'s non-directional transition-based formalism by adding three additional actions: SUBSTITUTE, DELETE, and INSERT. These new actions allow us to substitute, delete, or insert tokens during parsing sentences. Because the new actions may cause an infinite loop in derivation (e.g., infi-

nite substitutions and insertion-deletion alternations), I also introduce simple additional constraints that ensure the parser's termination. I evaluate the model with respect to accuracy and grammaticality improvements for ungrammatical sentences, demonstrating the scheme's robustness and applicability.

Chapter 5 and the following chapters consider "whole-sentence" GEC. First, chapter 5 describes issues in sentence-level GEC with respect to the evaluation metrics, datasets, and annotation scheme for the benchmark. As briefly explained in Section §1.3, the GEC community has mainly focused on improving and evaluating grammaticality on character and token-level errors such as spelling errors and closed-class errors (e.g., determiners, prepositions, verb-forms, subject-verb agreements, etc.). Also, the conventional annotation scheme relies on fine-grained error codes, which results in (1) very low inter-annotator agreement, even among experts, and (2) gold-standard edits that sound unnatural to native speakers of English. To address these issues, I introduce an alternate annotation scheme for GEC that considers *fluency* as well as *grammaticality* by asking whole-sentence rewrites instead of token- (or very short phrase) level edits with fine-grained error code constraints. I claim that the new annotation scheme has a stronger correlation with human judgment on widely used GEC metrics, which encourages a fundamental and necessary shift in the goal of GEC toward producing native fluency. I also explain an efficient way to collect human judgments to rank GEC systems via crowdsourcing. The method is based on a Bayesian online pairwise ranking aggregation called TrueSkill, and the technical details are described in **Appendix A**.

Chapter 6 presents a new dataset for benchmarking, **J**HU **FL**uency-**E**xtended **G**UG corpus (JFLEG), following the suggestions in Chapter 5. Compared with other GEC corpora, JFLEG includes a broad range of learners' proficiency as well as whole-sentence fluency edits. I also show the result of benchmark by four leading GEC systems on this dataset.

Chapter 7 proposes an end-to-end sentence-level GEC model that uses a neural encoder-decoder model with reinforcement learning. Unlike conventional maximum likelihood estimation (MLE), this model directly optimizes toward an objective that considers a sentence-level, task-specific evaluation metric, avoiding the exposure bias issue found in MLE. I demonstrate that the proposed model outperforms MLE in human and automated evaluation metrics, achieving the state of the art in a fluency-oriented GEC corpus.

Chapter 8, the final chapter, concludes the thesis and outlines ideas and suggestions for future research in GEC.

Chapter 2

Background

In this chapter, we will take a quick tour of the recent progress in grammatical error correction (GEC) in terms of the evaluation metrics, methods (i.e., models), and datasets for benchmarking. Although we start with studies from the 1980s, we primarily focus on related work published since 2000 indicating the issues that impede research progress in this area. As we saw in the introduction, the main theme here is the pivot of error types from character- and token-level error correction to whole-sentence error correction, namely grammaticality to fluency. This shift to whole-sentence error correction seems reasonable, but the extension of the scope has prompted some discrepancies regarding the appropriate benchmarking. This chapter attempts to make the issues clear and search for a solution to them. In the following sections, we first look at the evaluation metrics (§2.1), then move on to the GEC models (§2.2) and GEC corpora (§2.3).

2.1 Evaluation Metrics

Automated evaluation metrics are essential for keeping track of the progress of research. They enable us to compare performances among different models objectively. The objective comparison includes synchronic evaluation, as in shared task competitions, and diachronic evaluation, such as tracking the improvement of the state-of-the-art performance. Thus, it is very important to understand the evaluation metrics. They are something like a compass which should be shared and agreed on the research community. In other words, if the compass is not accurate, the community will be lost or waste time and labor in the wrong direction.

In the context of GEC, the task began from character and/or token-level error correction, in which the evaluation is simple and straightforward. The occurrence of these errors is typically measured by accuracy, precision, recall, and F-score (also called F-measure, F_1 score), which is the harmonic mean of precision and recall. Roughly speaking, they are computed by accumulating the number of binary (i.e., correct and incorrect) predictions.

Technically, given the following quadrants, each metric is calculated as follows:

	Positive (gold)	Negative (gold)
Positive (prediction)	True Positive (TP)	False Positive (FP)
Negative (prediction)	False Negative (FN)	True Negative (TN)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{F-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

In GEC, it is often very confusing to understand what "positive" and "negative" mean. The positive label means that the token has an error and should be corrected, while the negative label indicates that the token is correct and should be kept as it is. Namely,

- True Positive: the target token has an error, and the model makes the correct prediction.
- False Positive: the target token does not have an error, but the model makes a change incorrectly.
- False Negative: the target token has an error, but the model fails to detect it ("do-nothing").
- True Negative: the target token does not have an error, and the model does nothing.

In the history of GEC, these standard metrics have been used when the task was relatively simple, such as classification problems (e.g., correcting prepositions). For example,

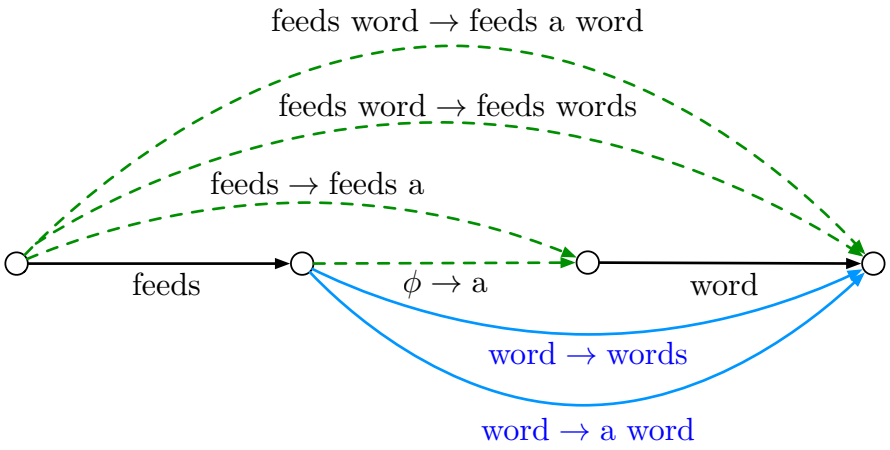

Figure 2.1: An example of the edit lattice created by token-based dynamic programming. The blue edges correspond to the (multiple) gold edits, and the green (dotted) edges are additional possible gold edits.

the HOO shared tasks used the F-score as the metric, in which they mainly targeted 6 types of errors: substitutions, deletions, and insertions of prepositions and determiners (Table1.1 in Chapter 1).

Following the success of the HOO shared tasks, the GEC community has conducted CoNLL shared tasks (Ng et al., 2013; Ng et al., 2014). With respect to the evaluation measure of the CoNLL shared tasks, MaxMatch (M^2) has been introduced (Dahlmeier and Ng, 2012b). M^2, a variant of F-score, was proposed to deal with both word- and phrase-level edits as well as allowing multiple references. M^2 can capture additional possible gold edits as well as the original gold edits by looking at the edit-lattice; let's take a look at the following example.

This example shows that the hypothesis is a correct edit but it is not credited by the mismatch with either gold edit. To address this issue, as shown in Figure 2.1, M^2 initially aligns a source and the hypothesis by token-based dynamic programming (i.e., edit-distance

Source:	Our baseline system feeds word into PB-SMT pipeline.
Reference 1:	Our baseline system feeds *a word* into PB-SMT pipeline.
Reference 2:	Our baseline system feeds *words* into PB-SMT pipeline.
Gold edit 1:	word / a word
Gold edit 2:	word / words
Hypothesis:	ϕ / a

measure), and then builds an edit-lattice that contains additional possible edits. In order to avoid unnecessarily long edits (e.g., baseline system feeds word / baseline system feeds *a* word), M^2 sets the maximum span as 2.) In the edit-lattice, the same edge cost is assigned except the original gold edits where negative cost is assigned. The shortest path algorithm is applied to extract the most reasonable set of edits between the hypothesis and reference.

M^2 was used as the official metric in CoNLL shared tasks and became a *de facto* standard metric for GEC. However, some limitations have been pointed out. First, phrase-level edits in M^2 can be easily gamed because the lattice treats a long phrase deletion as one edit. For example, when we put a single character 'X' as system output for each sentence, we obtain $P = 27.6, R = 29.5$, and $M^2 = 28.0$ (Table 2.1), which would be ranked 6th out of 13 systems in the CoNLL 2014 shared task. Another issue is that M^2 does not capture the difference between the "do nothing" baseline and "all wrong edits." In other words, the unchanged source sentences will get the lowest score of 0.0, and there is no penalty for making the source sentences *worse* (i.e., more ungrammatical).

Considering these problems, two other metrics have been proposed. Felice and Briscoe (2015) proposed I-Measure (IM), which computes the weighted accuracy of a token-level alignment between the source, hypothesis, and reference sentences. Although the IM con-

System	GLEU [0,100]	IM [-100,100]	M^2 [0, 100] P	R	$F_{0.5}$
"a"	0.2	0.0	28.4	31.3	28.9
"the"	0.2	-0.91	27.0	28.3	27.2
"."	0.2	-0.89	28.0	29.2	28.2
"x"	0.2	-0.93	27.6	29.5	28.0
"a a"	0.6	0.0	28.7	31.8	29.3
"the the"	0.6	-0.91	26.2	26.8	26.3
"a a a"	1.6	0.0	28.7	32.0	29.4
"the the the"	1.7	-0.90	25.1	26.3	25.4
Source	57.4	0.0	100.0	0.0	0.0
CAMB14	64.3	-5.3	39.7	30.1	37.3
CUUI14	64.6	-2.2	41.8	24.9	36.8
AMU14	64.6	-2.5	41.6	21.4	35.0
Src>Game	✓	✗	✓	✗	✗
Src<System	✓	✗	✗	✓	✓

Table 2.1: Metric scores of three artificially contrived systems (Game), input source sentences (Src), and top three system outputs, CAMB14 (Felice et al., 2014), CUUI14 (Rozovskaya et al., 2014), and AMU14 (Junczys-Dowmunt and Grundkiewicz, 2014) on CoNLL 2014 data (Ng et al., 2014). The bottom two rows show whether each metric scores the systems better than Game or worse than Source. Humans judge all systems be better than over Source.

siders the distinction between "do-nothing (already grammatical) baseline" and systems that only propose wrong corrections (i.e., make the source sentence worse), Grundkiewicz, Junczys-Dowmunt, and Gillian (2015) and Napoles et al. (2015) showed the *negative* correlation of the IM and human judgments. They also present that BLEU metric also has negative correlations when used for evaluating GEC systems. This is replicated more recently by (Choshen and Abend, 2018a). Furthermore, as Table 2.1 shows, IM is also easily gamed by dummy systems.

To address these issues, we proposed GLEU (Napoles et al., 2015), which is a variant of BLEU in machine translation. Based on BLEU (Papineni et al., 2002), GLEU computes *n*-

gram precision of the system output against reference sentences and additionally penalizes *n*-grams in the hypothesis that should have been corrected but failed. Formally,

$$\text{GLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{4} \frac{1}{n} \log p'_n\right)$$

$$p'_n = \frac{N(H,R) - [N(H,S) - N(H,S,R)]}{N(H)}$$

$$\text{BP} = \begin{cases} 1 & \text{if } h > r \\ \exp(1 - r/h) & \text{if } h \leq r \end{cases}$$

where $N(A, B, C, ...)$ is the number of overlapped n-grams among the sets, and BP *brevity penalty* is computed based on token length in the system hypothesis (h) and the reference (r). Similar to BLEU, GLEU computes *n*-gram precision between the system hypothesis (H) and the reference (R). In GLEU, however, *n*-grams in source (S) are also considered. The precision is penalized when the *n*-gram in H overlaps with the source and not with the reference. (i.e., phrases that should have been changed but weren't. This penalizes systems from doing nothing when they should have acted.) As Table 2.1 shows, GLEU addressed the above concerns (i.e., distinction between "do-nothing" and "all wrong edits," and robustness for gaming systems).

In this section, we have looked at the recent progress and issues in evaluation metrics for GEC. As mentioned above, evaluation metrics are important because they establish the research direction for the community. We will come back to the investigation of best practice for GEC benchmarking with respect to the evaluation metrics and dataset in Chapter 5.

2.2 Methods

In the GEC task, models are expected to detect and correct grammatical (and fluency) errors. While the scope of GEC has extended from closed-class error correction to whole-sentence correction, roughly four kinds of approaches have been proposed for GEC: the heuristic rules-based approach, classification (and the cascaded variant), phrase-based machine translation (PBMT), and neural machine translation (NMT). In this section, we will look at the pros and cons of each approach.

Rule-based approaches are effective when the errors depend on strict syntactic rules such as subject-verb agreement and noun numbers. The rule-based models have several advantages. First, they perform highly precisely; once the rule is detected, the correction is automatically applied. Another advantage of this approach is that it does not require a dataset to train the model, whereas recent statistical approaches, as we will see later on, need a large amount of (error-annotated or parallel) corpora. The rule-based approach is also helpful when it comes to giving feedback to the users (e.g., language learners,) by showing the applied rule explicitly. In fact, product-level systems such as ALEK (Assessing LExical Knowledge) at Educational Testing Service (Burstein and Chodorow, 1999; Chodorow and Leacock, 2000) and ESL Assistant from Microsoft (Gamon et al., 2009) employ rule-based approaches for specific types of errors.

One crucial drawback of the rule-based approach, however, is the coverage of error types. When it comes to grammatical errors that require larger contexts such as word choice, determiner, and pronoun errors, it is almost impossible to list all the heuristic rules.

For example, word choice and collocation rules are difficult to establish in advance, because learner errors are too diverse to be listed. Furthermore, the rule-based approach does not scale, because these rules are made by humans (often experts), which costs a huge amount of money and time.

As large numbers of text corpus have become available in the field, classification models have become very popular in the GEC community. The classification models are often used for closed-class errors such as preposition, determiner, and verb form errors. One unique aspect of this approach is feature designing (or feature engineering). In order to train classifiers, we are expected to decide what features will be useful for improving the performance. Typically, *n*-gram features are used as the baseline, and syntactic features (e.g., dependency labels) are added depending on the error type. For example, Lee (2004) proposed an article error correction system using a log-linear model (i.e., maximum entropy classifier) with syntactic features extracted from Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993). Nagata et al. (2006) used a log-linear model for mass/count noun prediction with more contextual features shared per discourse (i.e., beyond a sentence). Izumi, Uchimoto, and Isahara (2004) and Felice and Pulman (2008) applied a maximum entropy classifier for preposition errors. Dahlmeier and Ng (2011) proposed a joint article and preposition error correction model using an Alternating Structure Optimization (ASO) algorithm (Ando and Zhang, 2005). Rozovskaya and Roth (2011) applied various classification methods, such as Naïve Bayes (NB), Averaged Perceptron (AP), and Language Model (LM) score to establish out that NB performs the best for determiner and preposi-

tion errors if the prior of errors (i.e., confusion matrix) is adequately given. When it comes to correcting multiple error types, it is important to decide the order in which classifiers should be applied. Dahlmeier and Ng (2012a) proposed a beam search model in which multiple classifiers are iteratively ranked and selected to correct tokens one by one.

As the goal of GEC pivoted from closed-class error correction to "whole-sentence" correction, the classifier-based approaches faced several difficulties. First, new error types such as word choice, collocation/idiom errors, and tone (i.e., formality) errors, are not easily defined as a classification problem, because there are a too many possible classes (i.e., vocabulary size). In addition, L2 learners often make phrase-level errors, such as wrong phrase orders, that require a longer context (e.g., the sentence as a whole) to correct appropriately. The classification approach tends to use (or learn to use) the local features. Also, classifiers are applied independently for each error type, and they do not correct errors globally.

These issues may be addressed by the phrase-based machine translation (PBMT) technique (Koehn, 2009). In the PBMT approach, we treat the GEC task as a kind of machine translation problem, in which the input is an ungrammatical sentence and the errors are expected to be corrected in the output. Technically, the PBMT model defines the probability of a target (i.e., grammatical) sentence given a source (i.e., ungrammatical) sentence as follows:

$$\underset{e}{\operatorname{argmax}}\, p(e|f) = \underset{e}{\operatorname{argmax}}\, p(e)p(f|e),$$

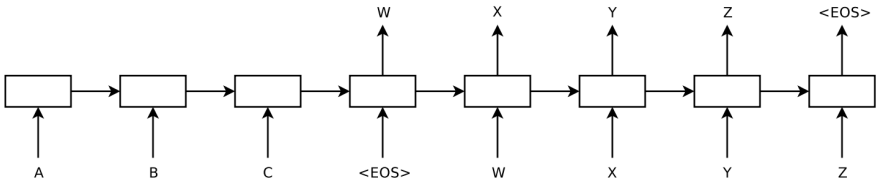

Figure 2.2: Illustrative example of the neural encoder-decoder model (adapted from (Sutskever, Vinyals, and Le, 2014)). The model encodes an input sentence "ACB" and produces "WXYZ" as the output.

where e is the grammatically correct hypotheses and f is the source (possibly ungrammatical) sentence to be corrected. Since it is almost intractable to collect all the possible f (i.e., all the possible ungrammatical sentences), the PBMT approach decomposes the problem into the combination of two simple problems: the English language model $p(e)$ and confusion probability $p(f|e)$. The English language model $p(e)$ is available from canonical English corpora (e.g., Wikipedia, newswire texts, etc.), and the confusion probability is obtained from existing GEC corpora.

Brockett, Dolan, and Gamon (2006) applied the PBMT approach to the GEC task for the first time, although they evaluated the model only on mass noun errors on the web. Park and Levy (2011) similarly developed a noisy channel approach for article, preposition, and verb form errors.

Most recently, along with the advance of neural network technique, neural Machine Translation (NMT) methods became popular. NMT is regarded as another whole-sentence rewriting method. Formally, the NMT model takes (possibly ungrammatical) source sentences $x \in X$ as an input, and predicts grammatical and fluent output sentences $y \in Y$ according to the model parameter θ (Sutskever, Vinyals, and Le, 2014) (Figure 2.2) The

model consists of two sub-modules, *encoder* and *decoder*. The encoder transforms x into a sequence of vector representations (hidden states) using a (bi)directional Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent neural network (GRU) (Chung et al., 2014). The decoder predicts a word y_t at a time, using previous token y_{t-1}. NMT is also able to handle all the error types (in principle) similarly to PBMT. Compared with the PBMT approach, NMT has the advantage of capturing long-term dependency in a sentence by the internal memory mechanism, such as the LSTM and GRU.

Yuan and Briscoe (2016) applied the NMT approach with an attention mechanism (Bahdanau, Cho, and Bengio, 2014), but it needs pre-processing and post-processing steps for replacing and aligning out-of-vocabulary (OOV) words, because GEC corpora contain a number of spelling errors. Chollampatt, Taghipour, and Ng (2016) used a phrase-based MT with two neural network features: a neural network global lexicon model (NNGLM) and a neural network joint model (NNJM). The NNGLM (Ha, Niehues, and Waibel, 2015) is a neural net that maps bag-of-words (in a sentence) into another bag-of-words representation, and the NNJM (Devlin et al., 2014) models the word probabilities for given source and target words. Schmaltz et al. (2016) demonstrate an attention-based encoder-decoder (sequence to sequence) model with convolutional neural nets (CNNs) for word representation. Although the model performs well, they tested it on a different data set from other GEC literature.

The main issue of PBMT and NMT is the requirement of a gigantic amount of parallel data to train. Compared to the machine translation community, parallel dataset for GEC is

Corpus	Size	Error-tagged	Public
Cambridge Learner Corpus (CLC)	200k texts	✓	✗
CLC-FCE (subset of CLC)	1,244 texts	✓	✓
Chinese Learners of English Corpus (CLEC)	1M words	✓	✗
Chungdahm corpus of English	131M words	✓	✗
English Taiwan Learner Corpus (ETLC)	5M words	partially	✗
HKUST corpus	30M words	✓	✗
International Corpus of Learner English (ICLE)	3M words	partially	✗
Lang-8 Learner Corpora	2.5M texts	partially	✓
NICT-JLE standard speaking test corpus	1.2M words	✓	✓
NUCLE	1M words	✓	✓

Table 2.2: GEC corpora and the basic information: 1. the size, 2. error-tagged (or edits by native speakers) or not, and 3. publicly and freely available or not.

very scarce, partly because the ESL learners' writing (e.g., essay) is often protected by the right to privacy.

In this section, we have presented major approaches in GEC and their pros and cons: the heuristic rules-based approach, classification (and the cascaded variants), phrase-based machine translation (PBMT), and neural machine translation (NMT). We will further examine the improvement of the GEC model in Chapter 7.

2.3 Datasets

In this section, we will look at the resources that are used for benchmarking GEC models. Overall, there are several corpora related to language learners, as shown in Leacock et al. (2014) and the Learner Corpora Around the World,[1] but most of them are "no-error

[1] https://uclouvain.be/en/research-institutes/ilc/cecl/learner-corpora-around-the-world.html

Corpus	Number of sentences	Number of references	Mean characters per sentence	Mean edit distance	Sentence changed	Error type labeld with the span	Diverse proficiency	Diverse topic	Diverse L1
NUCLE	59k	2	115	6	38%	✓	✗	✗	✗
FCE	34k	1	74	6	62%	✓	✓	✓	✓
Lang-8	2.5M	≥1	56	4	42%	✗	✓	✓	✓

Table 2.3: GEC corpora available for free (for research purposes) and several desired properties. ✓ and ✗ indicate whether the corpus exhibits each property.

tagged (or no reference)," "small scale," or publicly unavailable (Table 2.2).

Thus, in this section, we primarily focus on three publicly (and freely) available and error-tagged (i.e., edits by native speakers) GEC corpora: NUCLE, CLC-FCE, and Lang-8.[2]

The NUS Corpus of Learner English (NUCLE) is the most frequently used corpus for GEC. The NUCLE contains 1,414 essays (more than 1 million words) written by university students in Singapore (Dahlmeier, Ng, and Wu, 2013). Two language instructors coded each essay with 27 error codes.[3]

NUCLE was the official dataset of the 2013 and 2014 CoNLL shared tasks on GEC, and the 1,312 sentence test set from the 2014 task has become *de rigueur* for benchmarking GEC models. Before the NUCLE became available, there was no common dataset for

[2]We do not include the NICT-JLE corpus because the sources are obtained from speaking instead of writing.

[3]Versions 2.1 and later have 28 codes.

benchmarking. For example, Felice and Pulman (2008) use Cambridge Learner Corpus (CLC), (Gamon et al., 2008) uses Chinese Learners of English (CLEC), and (Tetreault and Chodorow, 2008) uses TOEFL essays. Importantly, this issue was not only the inconsistency of evaluation datasets; most of the GEC datasets were not publicly available.

The test set and system results from the most recent shared task were released to the community (Ng et al., 2014). The errors are tagged by two annotators and have been augmented with eight additional annotations from Bryant and Ng (2015). One of the drawbacks of NUCLE is the narrow diversity of proficiency, topics, and writers' native language (Table 2.3). In other words, the GEC models that are trained on this corpus are not likely to perform well on sentences in different topics, or sentences written by students with different proficiency levels. We will discuss this issue in more detail in Chapter 6.

The Cambridge Learner Corpus First Certificate in English (FCE), a freely and publicly available subset of the Cambridge Learner Corpus (CLC), has essays coded by one rater using about 80 error types alongside the score and demographic information (Yannakoudakis, Briscoe, and Medlock, 2011). The FCE was used for the Helping Our Own (HOO) 2012 shared task (Dale, Anisimoff, and Narroway, 2012), and Berzak et al. (2016) added dependency annotations to FCE. The FCE contains a broader representation of proficiency, topics, and native languages than NUCLE. However, the size is small and there is only one reference; corpus size is very important when we develop statistical models. In addition, there is no standard train/dev/test data split. These factors have made the FCE less popular than the NUCLE, particularly after the CoNLL shared tasks.

Error Type	%	Example sentence
Content word choice error	19.9	We need to deliver the merchandise on a daily *base/basis.
Preposition error	13.4	Our society is developing *in/at high speed.
Determiner error	11.7	We must try our best to avoid *the/a f fresh water.
Comma error	9.3	However, */, Ill meet you later.
Inflectional morphology	7.4	The women *weared/wore long dresses.
Wrong verb tense	6.7	I look forward to *see/seeing you.
Derivational morphology	4.9	It has already been *arrangement/arranged.
Pronoun	4.2	I want to make *me/myself fit.
Agreement error	4.0	I *were/was in my house.
Run-on sentence	4.0	The deliver documents to them they provide fast service.
Idiomatic collocation and word order	3.9	The latest issue *the magazine of/of the magazine ...
Confused words	1.9	I want to see the *personal/personnel manager.
Conjunction error	1.7	I want to see you *and/so that you can help me.
Words split with a space or joined	1.4	I organize sports *everyday/every day. It is also my *life style/lifestyle.
Apostrophe error (not including it/it's confusions)	1.3	We are all *sports/sports lovers.
Hyphenation error	1.3	It is a nourishing *low cost/low-cost meal.
Sentence fragment or two sentences that are joined	0.8	I'm going to get another one *. Because/because the old one broke.
Quantifier error	0.7	It doesn't give them too *much/many problems.
Other punctuation error	0.4	When are you leaving *./?
Negation formation	0.1	We *have not/do not have any time.

Table 2.4: Proportion of errors in Cambridge Learner Corpus (from Leacock et al. (2014)). Note: Spelling errors are excluded from this table.

Table 2.4 and Table 2.5 show the proportion of errors in the FCE and NUCLE. The statistics of error types are similar in both corpora. Based on the statistics, the shared tasks such as HOO and CoNLL 2013 focused initially on the following closed-class errors: determiner, preposition, noun number, verb form, and subject verb agreement. Also, these two corpora have been annotated with spans of text containing errors and assigned error codes. As we will see in Chapter 5, this style of error-coded annotation has some limitations with respect to cost, inter-annotator agreement rate, and fluency.

Error Type	%	Example
Article or determiner	14.6	It is obvious to see that [internet / the internet] saves people time and also connects people globally.
Wrong collocation/idiom	12.1	Early examination is [healthy / advisable] and will cast away unwanted doubts.
Redundancy	9.9	It is up to the [patient's own choice / patient] to disclose information.
Noun number	8.3	A carrier may consider not having any [child / children] after getting married.
Spelling, punctuation, capitalization	7.6	This knowledge [maybe relavant / may be relevant] to them.
Verb tense	6.9	Medical technology during that time [is / was] not advanced enough to cure him.
Preposition	6.0	This essay will [discuss about / discuss] whether a carrier should tell his relatives or not.
Word form	4.6	The sense of [guilty / guilt] can be more than expected.
Subject-verb agreement	3.5	The benefits of disclosing genetic risk information [outweighs / outweigh] the costs.
Verb form	3.3	A study in 2010 [shown / showed] that patients recover faster when surrounded by family members.
Other errors	3.1	An error that does not fit into any other category but can still be corrected.
Linking words/phrases	3.1	It is sometimes hard to find [out / out if] one has this disease.
Unclear meaning	2.4	Genetic disease has a close relationship with the born gene. (i.e., no correction possible without further clarification.)
Pronoun reference	2.3	It is everyones duty to ensure that [he or she / they] undergo regular health checks.
Run-on sentences, comma splices	1.8	The issue is highly [debatable, a / debatable. A] genetic risk could come from either side of the family.
Incorrect word order	1.5	[Someone having what kind of disease / What kind of disease someone has] is a matter of their own privacy.
Citation	1.3	Poor citation practice.
Tone (formal/informal)	1.2	[Its / It is] our family and relatives that bring us up.
Parallelism	1.1	We must pay attention to this information and [assisting / assist] those who are at risk.
Verb modal	1.0	Although the problem [would / may] not be serious, people [would / might] still be afraid.
Missing verb	1.0	However, there are also a great number of people [who / who are] against this technology.
Subordinate clause	0.9	This is an issue [needs / that needs] to be addressed.
Incorrect adjective/ adverb order	0.8	In conclusion, [personally I / I personally] feel that it is important to tell ones family members.
Noun possessive	0.5	Someone should tell the [carriers / carriers] relatives about the genetic problem.
Sentence fragment	0.5	However, from the ethical point of view.
Pronoun form	0.5	A couple should run a few tests to see if [their / they] have any genetic diseases beforehand.
Dangling modifiers	0.1	[Undeniable, / It is undeniable that] it becomes addictive when we spend more time socializing virtually.
Acronyms	0.1	After [WOWII / World War II], the population of China decreased rapidly.

Table 2.5: Proportion of errors in the NUS Corpus of Learner English (NUCLE) (from Ng et al. (2014)).

Unlike these, the Lang-8 Learner Corpus of Learner English (Tajiri, Komachi, and Matsumoto, 2012) is a parallel set of original and corrected sentences from `lang-8.com`,[4] an online community of language learners who post text to be corrected by other users. The parallel sentences are extracted by automatic alignment between the source and user-provided corrections. Unlike the NUCLE and FCE, Lang-8 does not require users to annotate error labels. Instead, users simply rewrite the original sentences so that they sound more natural to native speakers. In other words, users do not worry about detailed annotation guidelines that would include a number of error labels. As a result, Lang-8 corpus becomes the largest publicly (and freely) available resource for GEC, with more than 2 million English sentences (Table 2.3).[5] We discuss pros and cons of the different annotation procedures in Chapter 5.

2.4 Summary

In this chapter, we have looked at the recent progress in grammatical error correction (GEC) in terms of the evaluation metrics, methods, and corpora for benchmarking. The HOO and CoNLL shared tasks have introduced the importance of evaluation metrics and benchmarking dataset to the community. As the community grows and the scope of error types has extended from token-level to whole-sentence error correction, new datasets and various metrics have been proposed. We will discuss more details about the best practice

[4] As of August 2018, new signups for Lang-8 are suspended.

[5] Because of noise and implementation differences in sentence extraction, the size varies from 1–2.5 million sentences.

(i.e., combination of metrics and dataset) for GEC evaluation framework in Chapter 5.

Chapter 3

Character-level Error Correction: Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network[1]

Before diving into whole-sentence error correction, we first look at character-level error correction, i.e., spelling correction. In many cases, GEC systems require spelling correction as a preprocessing step,[2] and GEC's performance depends heavily on the performance in spelling error correction. In this chapter, motivated by a psycholinguistics study, we pro-

[1]Much of this chapter was originally published in Sakaguchi et al. (2017). Parts of this manuscript are intentionally jumbled to demonstrate the robust word processing ability of you, the reader.

[2]It is in fact controversial to say spelling errors are ungrammatical.

pose a spelling correction model that uses a semi character-level recurrent neural network.

3.1 Introduction

Despite the rapid improvement in natural language processing by computers, humans still have advantages in situations where the text contains noise. For example, the following sentences, introduced by a psycholinguist (Davis, 2003), provide a great demonstration of the robust word recognition mechanism in humans.

> *Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.*

To show an explicit contrast, let's take a look at the result by one of the commonly used commercial spelling checkers, where remaining errors (i.e., tokens that are failed to be corrected) are underlined.

> *Occurring to a scholarch at Cambridge Inertias , it does n't matter in what order the letters in a word are , the only impotent thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it outhit problem . This is bcuseae the human mind does not read every letter by istle , but the word as a whole .*

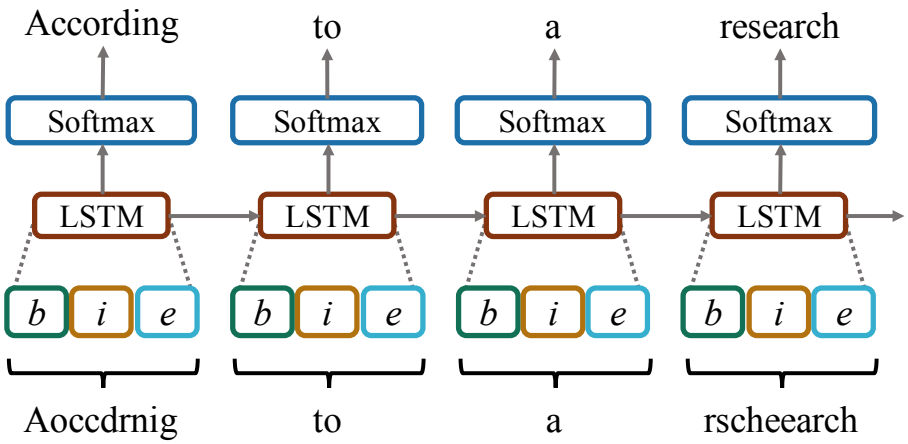

Figure 3.1: Schematic Illustration of semi-character recurrent neural network (scRNN).

This example shows the *Cmabrigde Uinervtisy* (*Cambridge University*) effect (or typoglycemia), which demonstrates that human reading is resilient to (particularly internal) letter transposition.

Robustness is an important and useful property for various tasks in natural language processing, and we propose a computational model which replicates this robust word recognition mechanism. The model is based on a standard recurrent neural network (RNN) with a memory cell as in long short-term memory (Hochreiter and Schmidhuber, 1997). We use an RNN because it has shown to be state-of-the-art language modeling (Mikolov et al., 2010) and it is also flexible to realize the findings from the *Cmabrigde Uinervtisy* effect. Technically, the input layer of our model consists of three sub-vectors: beginning (b), internal (i), and ending (e) character(s) of the input word (Figure 3.1). This semi-character level recurrent neural network is referred as scRNN.

First, we review previous work on the robust word recognition mechanism from psycholinguistics literature. Next, we describe technical details of scRNN which capture the

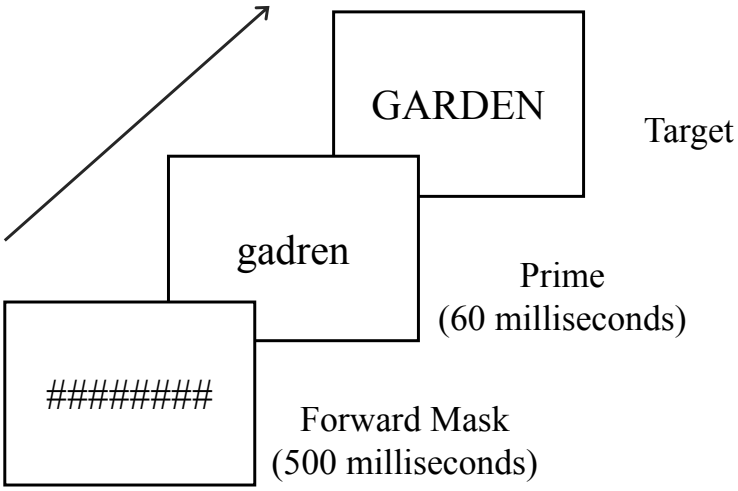

Figure 3.2: Example of the masked priming procedure.

robust human mechanism using recent developments in neural networks. As closely related work, we explain character-based convolutional neural network (CharCNN) proposed by Kim et al. (2015). Our experiments show that the scRNN significantly outperforms commonly used spelling checkers and CharCNN by (at least) 42% for jumbled word correction and 3% and 14% in other noise types (insertion and deletion). We also show that scRNN replicates recent findings from psycholinguistics experiments on reading difficulty depending on the position of jumbled letters, which indicates that scRNN successfully mimics (at least a part of) the robust word recognition mechanism by humans.

3.2 Raeding Wrods with Jumbled Lettres

Sentence processing with jumbled words has been a major research topic in psycholinguistics literature. One popular experimental paradigm is *masked priming*, in which a

Cond.	Example	# of fixations	Regression(%)	Avg. Fixation (ms)
N	The boy could not solve the problem so he asked for help.	10.4	15.0	236
INT	The boy cuold not slove the probelm so he aksed for help.	11.4*	17.6*	244*
END	The boy coudl not solev the problme so he askde for help.	12.6†	17.5*	246*
BEG	The boy oculd not oslve the rpoblem so he saked for help.	13.0‡	21.5†	259†

Table 3.1: Example sentences and results for measures of fixation excerpt from Rayner et al. (2006). There are 4 conditions: N = normal text; INT = internally jumbled letters; END = letters at word endings are jumbled; BEG = letters at word beginnings are jumbled. Entries with * have statistically significant difference from the condition N ($p < 0.01$) and those with † and ‡ differ from * and † with $p < 0.01$ respectively.

(lower-cased) stimulus, called *prime*, is presented for a short duration (e.g., 60 milliseconds) followed by the (upper-cased) target word, and participants are asked to judge whether the target word exists in English as quickly as possible (Figure 3.2).[3] The prime is consciously imperceptible due to the instantaneous presentation but it proceeds to visual word recognition by participants. The masked priming paradigm allows us to investigate the machinery of lexical processing and the effect of prime in a pure manner.

Forster et al. (1987) show that a jumbled word (e.g., gadren-GARDEN) facilitates primes as large as identity primes (garden-GARDEN) and these results have been confirmed in cases where the transposed letters are not adjacent (caniso-CASINO) (Perea and Lupker, 2004) and even more extreme cases (sdiwelak-SIDEWALK) (Guerrera and Forster, 2008).

These findings about robust word processing mechanism by humans have been further

[3]There is another variant for masked priming technique, where backward mask is inserted between the prime and target in addition to the forward mask.

investigated by looking at other types of noise in addition to simple letter transpositions. Humphreys, Evett, and Quinlan (1990) show that deleting a letter in a word still produces significant priming effect (e.g., blck-BLACK), and similar results have been shown in other research (Peressotti and Grainger, 1999; Grainger et al., 2006). Van Assche and Grainger (2006) demonstrate that a priming effect remains when inserting a character into a word (e.g., juastice-JUSTICE).

Another popular experimental paradigm in psycholinguistics is *eye-movement* tracking. In comparison to the masked priming technique, eye-movement paradigm provides data from normal reading process by participants. Regarding word recognition, the eye-tracking method has shown the relationship between a word difficulty and the eye fixation time on the word: when a word is difficult to process, average time to fixation becomes long. In addition, words that are difficult to process often induce regressions to words previously read.

With the eye-movement paradigm, Rayner et al. (2006) and Johnson, Perea, and Rayner (2007) conduct detailed experiments on the robust word recognition mechanism with jumbled letters. They show that letter transposition affects fixation time measures during reading depending on which part of the word is jumbled. Table 3.1 presents the result from Rayner et al. (2006). It is obvious that people can read smoothly (i.e., smaller number of fixations, regression, and average of fixation duration) when a given sentence has no noise (referring to this condition as N). When the characters at the beginning of words are jumbled (referring to this condition as BEG), participants have more difficulty (e.g.,

longer fixation time). The other two conditions, where words are internally jumbled (INT) or letters at word endings are jumbled (END), have similar amount of effect, although the number of fixations between them showed a statistically significant difference ($p < 0.01$). In short, the reading difficulty with different jumble conditions is summarized as follows: N $<$ INT $\leq$ END $<$ BEG.

It may be surprising that there is statistically significant difference between END and BEG conditions despite the difference being very subtle (i.e., fixing either the first or the last character). This result demonstrates the importance of beginning letters for human word recognition.[4]

3.3 Semi-Character Recurrent Neural Network

In order to achieve the human-like robust word processing mechanism, we propose a semi-character based recurrent neural network (scRNN). The model takes a semi-character vector (x) for a given jumbled word, and predicts a (correctly spelled) word (y) at each time step. The structure of scRNN is based on a standard recurrent neural network, where current input (x) and previous information is connected through hidden states (h) by applying a certain (e.g., sigmoid) function (g) with linear transformation parameters (W) and the bias (b) at each time step (t).

One critical issue of vanilla recurrent neural networks is that it is unable to learn long

[4]While there is still ongoing debate in the psycholinguistics community as to exactly how (little) the order of internal letters matter, here we follow the formulation of Rayner et al. (2006), considering only the letter order distinctions of BEG, INT, and END.

distance dependency in the inputs due to the vanishing gradient (Bengio, Simard, and Frasconi, 1994). To address the problem, Hochreiter and Schmidhuber (1997) introduced long short-term memory (LSTM), which is able to learn long-term dependencies by adding a memory cell (c). The memory cell has an ability to discard or keep previous information in its state. Technically, the LSTM architecture is given by the following equations,

$$i_n = \sigma\left(W_i\left[h_{n-1}, x_n\right] + b_i\right) \tag{3.1}$$

$$f_n = \sigma\left(W_f\left[h_{n-1}, x_n\right] + b_f\right) \tag{3.2}$$

$$o_n = \sigma\left(W_o\left[h_{n-1}, x_n\right] + b_o\right) \tag{3.3}$$

$$g_n = \sigma\left(W_g\left[h_{n-1}, x_n\right] + b_g\right) \tag{3.4}$$

$$c_n = f_n \odot c_{n-1} + i_n \odot g_n \tag{3.5}$$

$$h_n = o_n \odot \tanh\left(c_n\right) \tag{3.6}$$

where σ is the (element-wise) sigmoid function and $\odot$ is the element-wise multiplication.

While a standard input vector for RNN derives from either a word or a character, the input vector in scRNN consists of three sub-vectors (b_n, i_n, e_n) that correspond to the characters' position.

$$x_n = \begin{bmatrix} b_n \\ i_n \\ e_n \end{bmatrix} \tag{3.7}$$

The first and third sub-vectors (b_n, e_n) represent the first and last character of the n-th word. These two sub-vectors are therefore one-hot representations. The second sub-vector (i_n) represents a bag of characters of the word without the initial and final positions. For example, the word "University" is represented as $b_n = \{U = 1\}$, $e_n = \{y = 1\}$, and $i_n = \{e = 1, i = 2, n = 1, s = 1, r = 1, t = 1, v = 1\}$, with all the other elements being zero. The size of sub-vectors (b_n, i_n, e_n) is equal to the number of characters (N) in our language, and x_n has therefore the size of $3N$ by concatenating the sub-vectors.

Regarding the final output (i.e., predicted word y_n), the hidden state vector (h_n) of the LSTM is taken as input to the following softmax function layer with a fixed vocabulary size (v).

$$y_n = \frac{\exp\left(W_h \cdot h_n\right)}{\sum_v \exp\left(W_h \cdot h_n\right)} \tag{3.8}$$

We use the cross-entropy training criterion applied to the output layer as in most LSTM language modeling works; the model learns the weight matrices (W) to maximize the likelihood of the training data. This should approximately correlate with maximizing the number of exact word match in the predicted outputs. Figure 3.1 shows a pictorial overview of scRNN.

In order to check if the scRNN can recognize the jumbled words correctly, we test it in spelling correction experiments. If the hypothesis about the robust word processing mechanism is correct, scRNN will also be able to read sentences with jumbled words robustly.

3.4 Character-based Neural Network

Another possible approach to deal with reading jumbled words by neural networks is (pure) character-level neural network (Sutskever, Martens, and Hinton, 2011), where both input and output are characters instead of words. The character-based neural networks have been investigated and used for a variety of NLP tasks such as segmentation (Chrupala, 2013), dependency parsing (Ballesteros, Dyer, and Smith, 2015), machine translation (Ling et al., 2015), and text normalization (Chrupała, 2014).

For spelling correction, Schmaltz et al. (2016) uses character-level convolutional neural networks (CharCNN) proposed by Kim et al. (2015), in which the input is character but the prediction is at the word-level. More technically, according to Kim et al. (2015), CharCNN concatenates the character embedding vectors into a matrix $P_n \in \mathbb{R}^{d \times l}$ whose k-th column corresponds to the k-th character embedding vector (size of d) of n-th word which contains l characters. A narrow convolution is applied between P and *filter* $H \in \mathbb{R}^{d \times w}$ of width w, and then *feature map* $f_n \in \mathbb{R}^{l-w+1}$ is obtained by the following transformation[5] with a bias b.

$$f_n = \tanh(\mathbf{Tr}(P_n[:, k : k + w - 1]H^T) + b) \tag{3.9}$$

This is interpreted as a process of capturing important feature f with filter H to maximize

[5] In the equation, $P_n[:, k : k + w - 1]$ means the k-to-$(k + w - 1)$-th column of P_n.

the predicted word representation y_n by the *max-over-time*:

$$y_n = \max_k f_n[k] \tag{3.10}$$

Although CharCNN and scRNN have some similarity in terms of using a recurrent neural network, CharCNN is able to store richer representation than scRNN. In the following section, we compare the performance of CharCNN and scRNN with respect to jumbled word recognition task.

3.5 Experiments

We conducted spelling correction experiments to judge how well scRNN can recognize noisy word sentences. In order to make the task more realistic, we tested three different noise types: *jumble*, *delete*, and *insert*, where the *jumble* changes the internal characters (e.g., Cambridge $\rightarrow$ Cmbarigde), *delete* randomly deletes one of the internal characters (Cambridge $\rightarrow$ Camridge), and *insert* randomly inserts an alphabet into an internal position (Cambridge $\rightarrow$ Cambpridge). None of the noise types change the first and last characters. We used Penn Treebank for training, tuning, and testing.[6]

The input layer of scRNN consists of a vector with length of 76 (A-Z, a-z and 24 symbol characters). The hidden layer units had size 650, and total vocabulary size was set

[6]Section 2-21 for training, 22 for tuning, and 23 for test `https://catalog.ldc.upenn.edu/ldc99t42`. The data includes 39,832 sentences in training set (898k/950k tokens are covered by the top 10k vocabulary), 1,700 sentences in the tuning set (coverage 38k/40k), and 2,416 sentences in test set (coverage 54k/56k).

to 10k. We applied one type of noise to every word, but words with numbers (e.g., 1980s) and short words (length ≤ 3) were not subjected to jumbling, and therefore these words were excluded in evaluation. We trained the model by running 5 epochs with (mini) batch size 20. We set the backpropagation through time (BPTT) parameter to 3: scRNN updates weights for previous two words (x_{n-2}, x_{n-1}) and the current word (x_n).

For comparison, we evaluated CharCNN on the same training data,[7] and also compared widely-used spelling checkers (Enchant,[8] Commercial A, and Commercial B[9]).

3.5.1 Spelling correction results

Table 3.2 presents example outputs for the *Cmabrigde Uinervtisy* sentence by each model.[10] It may be surprising that CharCNN performs poorly compared with other spelling checkers. This is probably because the CharCNN highly depends on the order of characters in the word and the transposed characters adversely affected the recognition performance. Enchant, Commercial A, and Commercial B tend to fail long word correction. This may be because these models are not designed for severely jumbled input but they are likely to depend on edit distance between the incorrect and correct words. While these existing models struggle with correcting *Cmabrigde Uinervtisy* sentence, we see that scRNN

[7]For CharCNN, we employed the codebase available at `https://github.com/yoonkim/lstm-char-cnn.git`

[8]`http://www.abisource.com/projects/enchant/`

[9]We anonymized the name of the commercial product.

[10]The *Cmabrigde Uinervtisy* sentences contains jumbling as well as deletion, insertion, and replacement of characters. Note that we used a single scRNN (Jumble), and didn't train scRNN separately for each error type in this example.

Original	Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy , it deos n't mttaer in waht oredr the ltteers in a wrod are , the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae . The rset can be a toatl mses and you can sitll raed it wouthit porbelm . Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef , but the wrod as a wlohe .
Correct	According to a researcher at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it without problem . This is because the human mind does not read every letter by itself , but the word as a whole .
CharCNN (Kim et al.)	According to a research at Cambridge Minority , it deck nt mother in wait or the letters in a wood are , the tony Vermont timing is taxi the tourist and sat letter be at the fruit pile . The reset can be a total uses and you can vital rake it worthy parallel . Mips is abuse the human trim deck not rake survey latter by leftist , but the wood as a whole .
Enchant	Ecuadoran to a searcher at Brigade Nerviness , it does n't matter in what order the letters in a word are , the only omnipresent thing is that the freest and slat letter be at the right place . The rest can be a total mess and you can still read it outhit corbel . Tish is Ceausescu , the human mind does not read Hervey letter by leftist , but the word as a whole .
Commercial A	Occurring to a scholarch at Cambridge Inertias , it does n't matter in what order the letters in a word are , the only impotent thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it outhit problem . This is bcuseae the human mind does not read every letter by istle , but the word as a whole .
Commercial B	Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy , it does n't matter in what order the letters in a word are , the only iprmoetnt thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it wouthit problem . Tihs is bcuseae the human mind does not read every letter by itself , but the word as a whole .
scRNN (proposed)	According to a research at Cambridge University , it does n't matter in what order the letters in a word are , the only important thing is that the first and last letter be at the right place . The rest can be a total mess and you can still read it without problem . This is because the human mind does not read every letter by itself , but the word as a whole .

Table 3.2: Example spelling correction outputs for the *Cmabrigde Uinervtisy* sentences. Words that the system failed to correct are underlined. CharCNN stands for the character-based convolutional neural network by Kim et al. (2015).

demonstrates significantly better recognition ability. The only error in scRNN may be because the last character (rscheearc**h**) activated the scRNN nodes strongly toward *research*

	Jumble	Delete	Insert
CharCNN (Kim et al.)	17.17	21.30	35.00
Enchant	57.15	37.01	88.54
scRNN (proposed)	**98.96**	**85.74**	**96.70**

Table 3.3: Spelling correction accuracy (%) with different error types on the entire test set.

	Jumble	Delete	Insert
CharCNN (Kim et al.)	16.18	19.76	35.53
Enchant	57.59	35.37	89.63
Commercial A	54.81	60.19	93.52
Commercial B	54.26	71.67	73.52
scRNN (proposed)	**99.44**	**85.56**	**97.04**

Table 3.4: Spelling correction accuracy (%) with different error types on the subset of test set (50 sentences).

instead of *researcher*.[11]

Table 3.3 and 3.4 show the overall result on the test set with respect to noise type. We also tested spelling correction on a small subset (50 sentences) because of the API limits etc. of commercial systems. Overall, as seen in the example above, scRNN significantly outperforms the other spelling checker models across all three noise types. Since scRNN is especially designed for jumbled word recognition, it is not surprising that it performs particularly well on *jumble* noise. However, it is striking that scRNN outperforms the other models in deletion and insertion errors as well.[12] This clearly demonstrates the robustness of scRNN.

The relatively large drop in *delete* in scRNN may be because the information lost by deleting character is significant. For example, when the word *place* has dropped the char-

[11] There is also an deletion of 'r'.

[12] It is important to note that some commercial systems have constraints of the model size (Church, Hart, and Gao, 2007).

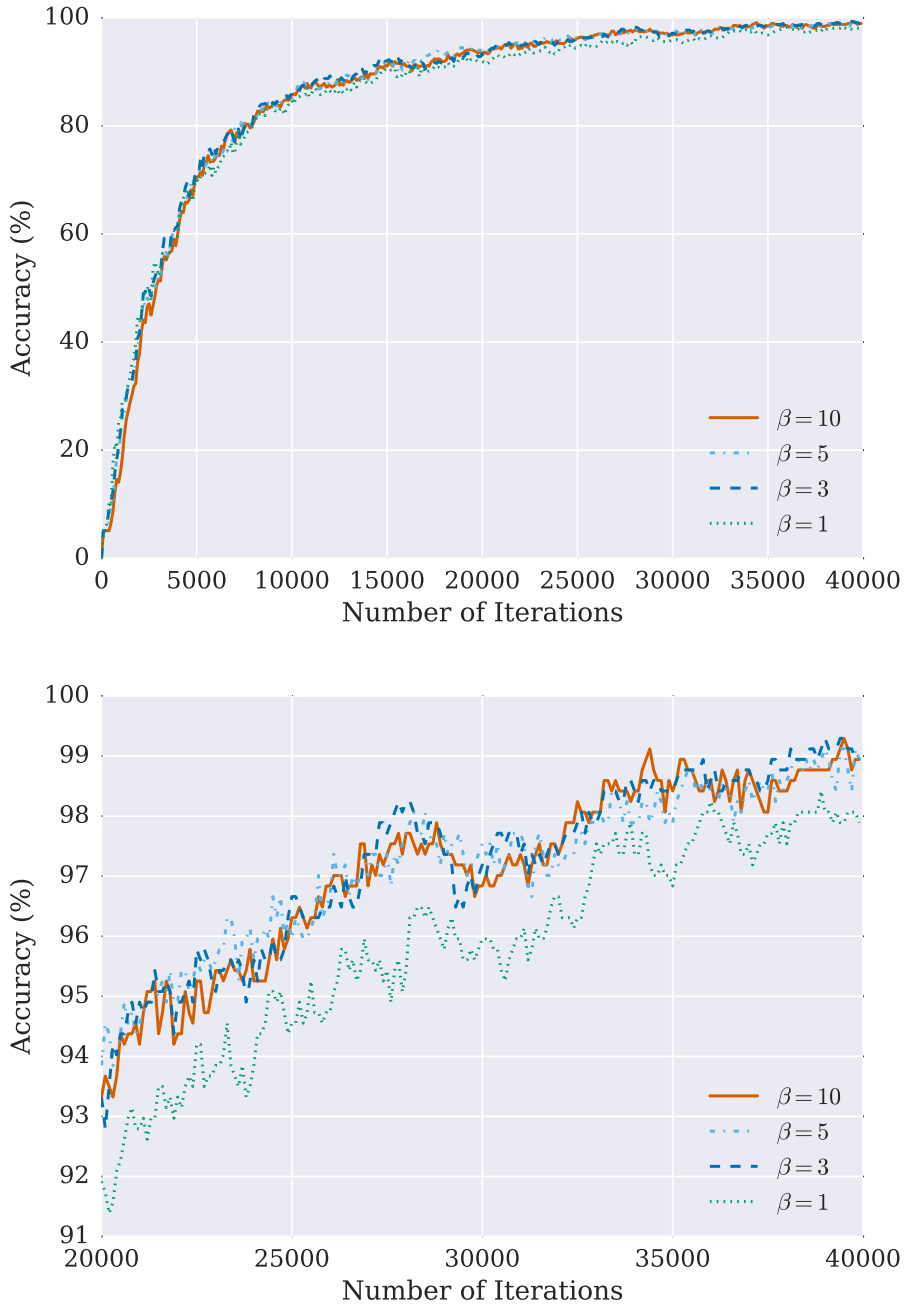

Figure 3.3: Learning curve of training scRNN with different BPTT parameter (on dev set): first 40k iterations (top) and its enlarged view between 20k and 40k iterations (bottom).

β	Accuracy (%)	SD
1	98.69	0.53
3	98.96	0.45
5	98.91	0.40
10	98.95	0.43

Table 3.5: scRNN accuracy (%) on jumbled word recognition with different BPTT parameters. There were no statistically significant differences among them.

Units	Acc (%)	SD	Size (KB)
5	24.65	2.59	236
10	48.43	3.26	435
15	73.32	3.65	632
20	84.82	2.39	830
30	94.15	1.54	1,255
40	96.90	1.26	1,670
50	98.48	0.94	2,092
60	98.39	0.81	2,514

Table 3.6: scRNN accuracy (%), the standard deviation, and the size of model file (KB) on jumbled word recognition with respect to the number of units of LSTM.

acter *l*, the surface form becomes *pace*, which is also a valid word. Also, the word *mess* with *e* being deleted produces the form of *mss*, which can be recovered as *mess*, *mass*, *miss*, etc. In the *Cmabrigde Uinervtisy* sentences, in both cases, the local context support other phrase such as 'at the right *pace/place*' and 'a total *mass/mess*'. These examples clearly demonstrate that deleting characters harm the word recognition more significantly than other noise types. All the models perform relatively well on *insert* noise, indicating that adding extraneous information by inserting a letter does not change the original information significantly.

With respect to a learning curve on scRNN (Figure 3.3, top), we found that the model achieves 0.9 (in accuracy) at the 15,000-th iteration of the mini batch. This can be made

Cond.	Example	Accuracy
INT	As a relust , the lnik beewetn the fureuts and sctok mretkas rpiped arapt .	98.96
END	As a rtelus , the lkni betwene the feturus and soctk msatrek rpepid atarp .	98.68*
BEG	As a lesurt , the lnik bweteen the utufers and tocsk makrtes pipred arpat .	98.12†
ALL	As a strule , the lnik eewtneb the eftusur and okcst msretak ipdepr prtaa .	96.79‡

Table 3.7: Example sentences and results for spelling correction accuracy by scRNN variants depending on different jumble conditions: INT = internal letters are jumbled; END = letters at word endings are jumbled; BEG = letters at word beginnings are jumbled; ALL = all letters are jumbled. Entries with * have a difference with marginal significance from the condition INT ($p = 0.07$) and those with † and ‡ differ from * and † with $p < 0.01$ respectively.

Cond.	Examples of errors (correct/wrong)
INT	Once/once, Under/under, Also/also, there/three, form/from, fares/fears, trail/trial, Broad/Board
END	being/begin, quiet/quite, bets/best, stayed/steady, heat/hate, lost/lots + same errors in INT
BEG	Several/reveal, Growth/worth, host/shot, creditors/directors, views/wives + same errors in INT
ALL	Under/trend, center/recent, licensed/declines, stop/tops + same errors in INT, END, & BEG

Table 3.8: Error analysis of scRNN variants.

within a hour with a CPU machine, which demonstrates simplicity of scRNN compared with CharCNN.

Figure 3.3 also shows the effect of BPTT size (β), and the accuracy on test set is presented in Table 3.5. As explained, β indicates the context length of updates during training. It is surprising that the longer contexts ($\beta = 5, 10$) do not necessarily yield better performance. This is probably because it rarely happens that the context plays an important role on distinguishing ambiguous representation (e.g., anagrams) in scRNN. If we take closer look at the learning curve (Table 3.3, bottom), however, there is a clear gap in learning

efficiency between with and without contexts (i.e., β=1 vs. the rest).

Finally, we reduced the number of units in the hidden layer to see the model size and performance of scRNN. Surprisingly, as Table 3.6 presents, scRNN with 50 units already achieves comparable results to 650 units (Table 3.3). The result suggests that 50 units (2 MB) are enough to distinguish 10k English words.

3.5.2 Corroboration with psycholinguistic experiments

As seen in the literature review in psycholinguistics, the position of jumbled characters affects the cognitive load of human word recognition. We investigate this phenomenon with scRNN by manipulating the structure of input vector. We replicate the experimental paradigm in Rayner et al. (2006), but using scRNN rather than human subjects. We trained scRNN variants depending on different jumble conditions: INT, END, BEG, and ALL. INT is the same model as explained in the previous section ($x_n = [b_n, i_n, e_n]^T$). END represents an input word as a concatenation of the initial character vector (b) and a vector for the rest of characters ($x_n = [b_n, i_n + e_n]^T$). In other words, in END model, the internal and last characters are subject to jumbling. BEG model combines a vector for the final character (e) and a vector for the rest of characters ($x_n = [b_n + i_n, e_n]^T$). In other words, initial and internal characters are subject to jumbling. In ALL model, all the letters are subject to jumble (e.g., research vs. eesrhrca) and represented as a single vector ($x_n = [b_n+i_n+e_n]^T$). This is exactly the same as bag of characters. We trained all the scRNN variants with $\beta = 3$, the number of hidden layer units being 650, and total vocabulary size to be 10k.

Table 3.7 shows the result. While all the variants of scRNN achieve high accuracy, the statistical test revealed that INT and END have a difference with statistically marginal significance ($p = 0.07$). There are statistically significant differences ($p < 0.01$) both in END&BEG and BEG&ALL. From the results, the word recognition difficulty of different jumbled types is summarized as INT $\leq$ END $<$ BEG $<$ ALL, which is the same order as the finding in Table 3.1. It is not surprising that INT outperform the other variants because it has richer representation in x_n (twice or three times larger than the other variants). However, it is interesting to see that END outperforms BEG both in Rayner et al. (2006) and our experiment despite that the size of x_n between END and BEG models are equal. This suggests the scRNN replicates (at least a part of) the human word recognition mechanism, in which the first letter is more important and informative than the last one in English.

For qualitative analysis, Table 3.8 shows some errors (correct/wrong) that each variant made. All the scRNN variants often fail to recognize capitalized first character (e.g., Once/once, Under/under), specifically when the word is at the beginning of the sentence. Other than the capitalization errors, most errors come from anagrams. For example, errors in INT (the original scRNN) are internally anagrammable words (e.g., there/three, form/from). END model made errors on words that are anagrammable with the first character being fixed (e.g., being/begin, quiet/quite). BEG model, on the other hand, failed to recognize anagrammable words with the last character being the same (e.g., creditors vs. directors, views vs. wives). In addition, BEG model often ignores the first (uppercased) character of the word (e.g., Several/reveal, Growth/worth). Finally, ALL model

failed to recognize anagrammable words (e.g., center/recent, licensed vs. declines). Although scRNN generally disambiguated anagrammable words successfully from context, all these examples from the error analysis are straightforward and convincing when we consider the characteristics of each variant of scRNN.

3.6 Summary

In this chapter, we looked at character-level error correction as the first step toward sentence-level error correction. We have presented a semi-character recurrent neural network model, scRNN, which is inspired by the robust word recognition mechanism known in psycholinguistics literature as the *Cmabrigde Uinervtisy* effect (or typoglycemia). Despite the model's simplicity compared to character-based convolutional neural networks (CharCNN), it significantly outperforms widely used spelling checkers with respect to various noise types. We also have demonstrated a similarity between scRNN and human word recognition mechanisms, by showing that scRNN replicates a psycholinguistics experiment about word recognition difficulty in terms of the position of jumbled characters.

There are a variety of potential NLP applications for scRNN where robustness plays an important role, such as normalizing social media text (e.g., *Coooooolll* $\rightarrow$ *Cool*), post-processing of OCR text, and modeling morphologically rich languages, which could be explored with this model in future work.

Of course, with respect to GEC, scRNN can be applied to it as the first preprocessing

step before running token-level, or phrase-level error correction algorithms.

In the next chapter, we move one step forward, i.e., token-level error correction.

Chapter 4

Token-level Error Correction: Error-repair Dependency Parsing for Ungrammatical Texts[1]

In this chapter, we move one step forward from character-level error correction. Specifically, this chapter is about token-level grammatical error correction. Token-level error correction often requires the sentence's syntactic information, but syntactic parsers fails to parse ungrammatical sentences because they are trained on grammatical sentences. The issue is like a chicken-and-egg problem; we want to parse ungrammatical sentences to correct the errors, but how can we parse sentences without knowing correct grammar? In this chapter, we examine a joint model where token-level error correction and dependency

[1] Much of this chapter was originally published in Sakaguchi, Post, and Van Durme (2017a).

parsing are executed jointly.

4.1 Introduction

Robustness has always been a desirable property for natural language parsers: humans generate a variety of noisy outputs, such as ungrammatical webpages, speech disfluencies, and the text in language learner's essays. Such *non-canonical* text contains grammatical errors such as substitutions, insertions, and deletions. For example, a non-native speaker of English might write *"*I look in forward hear from you"*, where *in* is inserted, *to* is deleted, and *hearing* is substituted incorrectly.

We propose a novel dependency parsing scheme that jointly parses and repairs ungrammatical sentences with these sorts of errors. The parser is based on the *non-directional easy-first* (EF) parser introduced by Goldberg and Elhadad (2010) (GE hereafter), which iteratively adds the most probable arc until the parse tree is completed. These actions are called ATTACHLEFT and ATTACHRIGHT depending on the direction of the arc. We extend the EF parsing scheme to be robust for ungrammatical inputs by correcting grammatical errors with three new actions: SUBSTITUTE, INSERT, and DELETE. These new actions do not add an arc between tokens but instead they edit a single token. As a result, the parser is able to jointly parse and correct grammatical errors in the input sentence. We call this new scheme *Error-Repair Non-Directional Easy-First parsing* (EREF). Since the new actions may greatly increase the search space (e.g., infinite substitutions), we also introduce simple

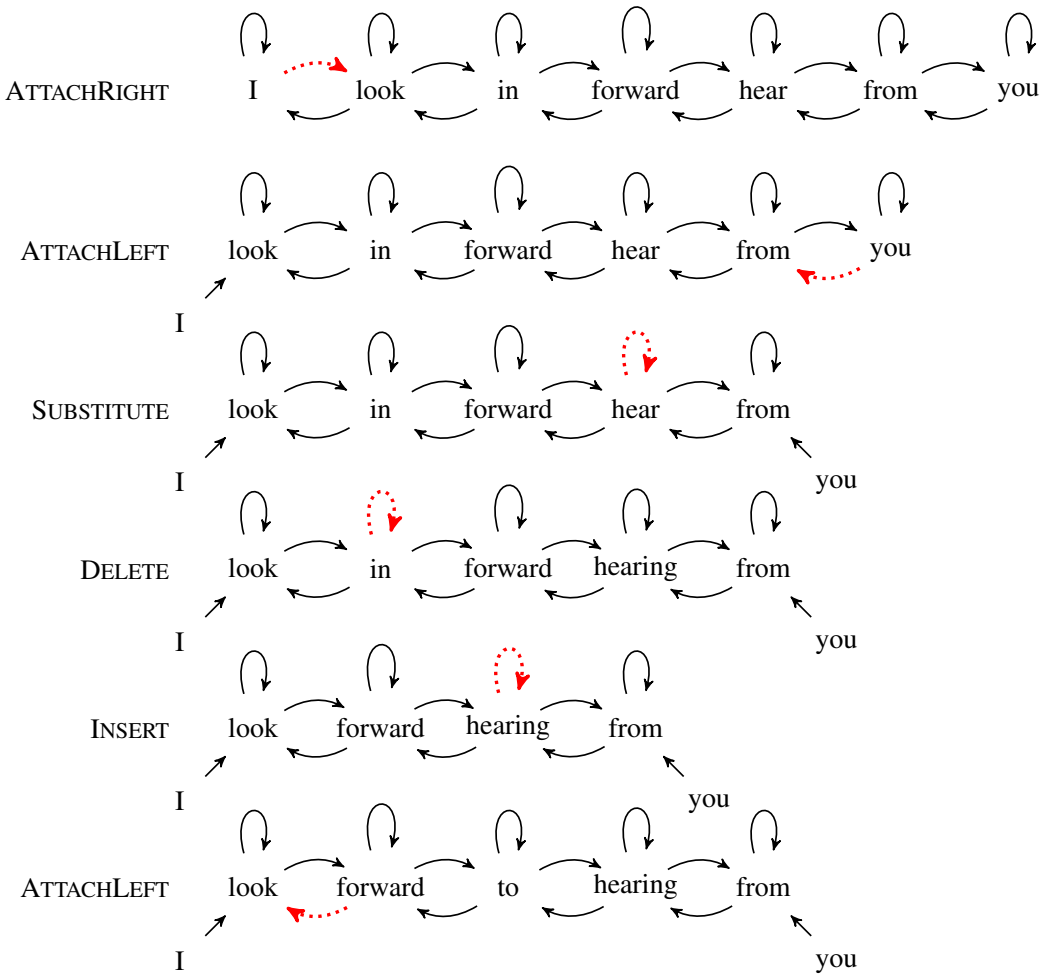

Figure 4.1: Illustrative example of partial derivation under error-repair easy-first non-directional dependency parsing. Solid arrows represent ATTACHRIGHT and ATTACHLEFT in Goldberg and Elhadad (2010). Dotted arcs correspond to actions for each step. Following the notation by GE, arcs are directed from a child to its parent.

constraints to avoid such issues.

We first describe the technical details of EREF (§7.2) and then evaluate our EREF parser with respect to dependency accuracy (robustness) and grammaticality improvements (§7.3). Finally, we position this effort at the intersection of noisy text parsing and grammatical error correction (§7.4).

4.2 Model

4.2.1 Non-directional Easy-first Parsing

Let us begin with a brief review of a non-directional easy-first (EF) parsing scheme proposed by GE, which is the foundation of our proposed scheme described in the following sections.

The EF parser has a list of partial structures p_1, ..., p_k (called *pending*) initialized with sentence tokens w_1, ..., w_n, and it keeps updating *pending* through derivations. Unlike left-to-right (e.g., shift-reduce) parsing algorithms (Yamada and Matsumoto, 2003; Nivre, 2004), EF iteratively selects the best pair of adjoining tokens and chooses the direction of attachment: ATTACHLEFT or ATTACHRIGHT. Once the action is committed, the corresponding dependency arc is added and the child token is removed from *pending*. The first two derivations in Figure 4.1 depict ATTACHRIGHT and ATTACHLEFT. Pseudocode is shown in Algorithm 1 (lines 1, 3-12).

The parser is trained using the structured perceptron (Collins, 2002) to choose actions to take given a set of features expanded from templates. The cost of actions is computed at every step by checking the *validity*: whether a new arc is included in the gold parse and whether the child already has all its children. See GE for further description of feature templates and structured perceptron training. Since it is possible that there are multiple *valid* sequence of actions and it is important to examine a large search space, the parser is allowed to explore (possibly incorrect) actions with a certain probability, termed *learning*

with exploration by Goldberg and Nivre (2013).

4.2.2 Error-repair variant of EF

Error-repair non-directional easy-first parsing scheme (EREF) is a variant of EF. We add three new actions: SUBSTITUTE, DELETE, INSERT as $Acts_{\text{ER}}$. We do not deal with a swapping action (Nivre, 2009) to deal with word reordering errors, since these errors and the swapping actions make the problem much harder. Thankfully, as (Leacock et al., 2014) reports, the word order errors are even less frequent than other error types. SUBSTITUTE replaces a token to a grammatically more probable token, DELETE removes an unnecessary token, and INSERT inserts a new token at a designated index. These actions are shown in Figure 4.1 and Algorithm 1 (lines 13-25). Because the length of *pending* decreases as an attachment occurs, the parser also keeps the token indices in *repaired* (line 5), which holds all tokens in a sentence throughout the parsing process. Furthermore, the parser updates token indices in *pending* and *repaired* when INSERT or DELETE occurs. Technically, when a token at i is deleted/inserted, the parser decrements/increments the indices that are $k >= i$ (before executing the action) in *pending*, *repaired*, and parents and children in a (partial) dependency tree (*Arcs*).

To find the best candidate for SUBSTITUTE and INSERT efficiently, we restrict candidates to the same part-of-speech or pre-defined candidate list. We select the best candidate by comparing each n-gram language model score with the same surrounding context.

Similar to EF, while training the parser, the cost for $Acts_{\text{ER}}$ is based on *validity*. The

Algorithm 1: Error-repair non-directional parsing

Input: ungrammatical sentence= w_1 ... w_n
Output: a set of dependency arcs (*Arcs*), repaired sentence ($\hat{w}_1$ ... $\hat{w}_m$)

Acts = { ATTACHLEFT, ATTACHRIGHT }
Acts$_{\text{ER}}$ = { DELETE, INSERT, SUBSTITUTE }
Arcs = { }
pending = $p_1 ... p_n \leftarrow w_1 ... w_n$
repaired = $\hat{w}_1 ... \hat{w}_n \leftarrow w_1 ... w_n$
while *len (pending)* > 1 **do**
    best $\leftarrow \underset{act \in Acts \cup Acts_{\text{ER}}}{\text{argmax}}$ *score* (*act* (i))
    s.t. $1 \leq i \leq len(pending)$ $\cap$ *isLegal*(*act*, *pending*)
    if *best*$\in$ *Acts* **then**
        (*parent*, *child*) $\leftarrow$ *edgeFor*(*best*)
        Arcs.add((*parent*, *child*))
        pending.remove(*child*)
    else if *best* = SUBSTITUTE **then**
        c = *bestCandidate*(*best*, *repaired*)
        pending.replace(p_i, c)
        repaired.replace($\hat{w}_{p_i.idx}$, c)
    else if *best* = DELETE **then**
        pending.remove(p_i)
        repaired.remove($\hat{w}_{p_i.idx}$)
        Arcs.updateIndex()
    else if *best* = INSERT **then**
        c = *bestCandidate*(*best*, *repaired*)
        pending.insert(i, c)
        repaired.insert($p_i.idx, c$)
        Arcs.updateIndex()
end
return *Arcs, repaired*

validity of the new actions is computed by taking the edit distance (*d*) between the *Gold* tokens (w_1^* ... w_r^*) and the sentence state that the parser stores in *repaired* ($\hat{w}_1$ ... $\hat{w}_m$). When the edit distance after taking an action (d_{after}) is smaller than before (d_{before}), we regard the action as *valid* (Algorithm 2).

One serious concern of EREF is that the new actions may cause an infinite loop during

parsing (e.g., infinite SUBSTITUTE, or an alternative DELETE and INSERT sequence.). To avoid this, we introduce two constraints: (1) *edit flag* and (2) *edit limit*. *Edit flag* is assigned for each token as a property, and a parser is not allowed to execute $Acts_{\mathrm{ER}}$ on a token if its flag is on. The flag is turned on when a parser executes $Acts_{\mathrm{ER}}$ on a token whose flag is off. In INSERT action, the flag of the inserted token is activated, while the subsequent token (which gave rise to the INSERT) is not. *Edit limit* is set to be the number of tokens in a sentence, and the parser is not allowed to perform $Acts_{\mathrm{ER}}$ when the total number of execution of $Acts_{\mathrm{ER}}$ exceeds the limit. These two constraints prevent the parser from falling into an infinite loop as well as parsing in the same order of time complexity as GE. We also add the following constraints to avoid unreasonable derivations: (i) a word with a dependent cannot be deleted and (ii) any dependent (i.e., already attached) tokens cannot be substituted. All the constraints are implemented in the *isLegal*() function in Algorithm 1 (line 8). The *isLegal*() function checks if the proposed action does not violate the new constraints to avoid unreasonable derivations. The *isValid*() function checks if the proposed action makes the current state closer to the gold (and grammatically correct) parse tree. (i.e., the right action). We note that these constraints not only prevent undesirable derivations but also leads to an efficiency in exploring the search space during training.

Algorithm 2: Check validity during training

```
1 Function isValid(act, repaired, Gold)
2     d_before = editDistance(repaired, Gold)
3     repaired+ = repaired.apply(act)
4     d_after = editDistance(repaired+, Gold)
5     if d_before > d_after then return true;
6     else return false;
7 end
```

4.3 Experiment

4.3.1 Data and Evaluation

We evaluate EREF with respect to dependency parsing accuracy (Experiment 1) and grammaticality improvement (Experiment 2).[2]

In the first experiment, as in GE, we train and evaluate our parser on the English dataset from the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993) with the Penn2Malt conversion program (Sections 2-21 for training, 22 for tuning, and 23 for test). We use the PTB for the dependency experiment, since there are no ungrammatical text corpora that has dependency annotation on the *corrected* texts by human.

We choose the following most frequent error types that are used in CoNLL 2013 shared task (Ng et al., 2013):

1. Determiner (substitution, deletion, insertion)

2. Preposition (substitution, deletion, insertion)

[2]Code for the experiments is available at `http://github.com/keisks/error-repair-parsing`

3. Noun number (singular vs. plural)

4. Verb form (tense and aspect)

5. Subject verb agreement

Regarding the candidate sets for INSERT and SUBSTITUTE actions, following Rozovskaya and Roth (2014), we focus on the most common candidates for each error type, setting the determiner candidates to be {*a, an, the,* ϕ (as deletion)}, preposition candidates to be {*on, about, from, for, of, to, at, in, with, by,* ϕ}, and verb forms to be VBP, VBZ, VBG, VBD, and VBN.[3] We build a 5-gram language model on English Gigaword with the KenLM Toolkit (Heafield, 2011) for EREF to select the best candidate.

We manually inject grammatical errors into PTB with certain error-rates similarly to the GenERRate toolkit by Foster and Andersen (2009), which is designed to create synthetic errors into sentences to improve grammatical error detection. More concretely, we decide the noise injection rate e. According to the noise injection rate, the tool randomly delete, insert, or replace tokens. For example, when the sentence has n tokens and $e = 0.1$ (i.e., 10%), noise are injected to about $0.1 \cdot n$ tokens. It is also possible to configure specific noise injection rates depending on the error type. We randomly select the noise type (e.g., delete, insert, or replace) for all the error types except preposition errors. For preposition errors, we use the confusion matrix extracted from Felice and Pulman (2008).

We train and tune EREF models with different token-level error injection rates from 5% (E05) to 20% (E20), because language learner corpora have generally around 5% to 15%

[3]We assume that the input contains the part of speech information.

of token level errors depending on learners' proficiency (Leacock et al., 2014). Since the error injection is stochastic, we train each model with 10 runs and take an average of parser performance on the test set.

As a baseline, we use the original parser as described by GE, which is equivalent to EREF with training on an error-free corpus (E00). Since the EF baseline does not allow error correction during parsing, we pre-process the test data with a grammatical error correction system similar to Rozovskaya and Roth (2014), where a combination of classifiers for each error type corrects grammatical errors.

For evaluation, we jointly parse and correct grammatical errors in the test set with different error injection rates (from 0% to 20%). It is important to note that the number of tokens between the parser output and the oracle may be different because of error injection into the test set and $Acts_{\mathrm{ER}}$ during parsing. To handle this mismatch, we evaluate the dependency accuracy with alignment (Favre, Bohnet, and Hakkani-Tür, 2010) in the spirit of SParseval (Roark et al., 2006), where tokens between a hypothesis and oracle are aligned prior to calculating the dependency accuracy.

In the second experiment, we use the Treebank of Learner English (TLE) (Berzak et al., 2016) to see the grammaticality improvement in a real scenario. TLE contains 5,124 sentences and 2.69 (std 1.9) token errors per sentence. The average sentence length is 19.06 (std 9.47). TLE also provides dependency labels and POS tags on the *raw (ungrammatical)* sentences. It is important to note that TLE has dependency annotation only for the original *ungrammatical* sentences, and therefore we do not compute the accuracy of dependency

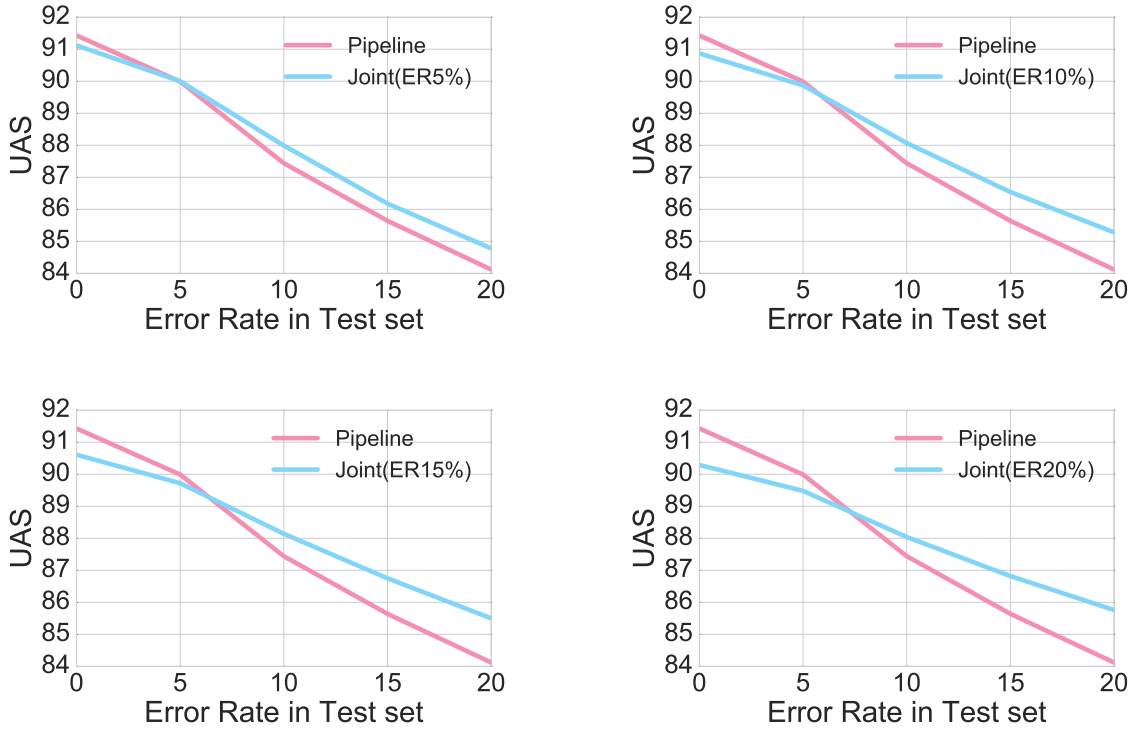

Figure 4.2: Unlabeled dependency accuracy results with the 5x5 models and test sets (higher is better).

parse in this experiment. Since the corpus size is small, we train EREF (E05 to E20) on 100k sentences from Annotated Gigaword (Napoles, Gormley, and Van Durme, 2012) and used TLE as a test set. Spelling errors are ignored because EREF can use the POS information. Grammaticality is evaluated by a regression model (Heilman et al., 2014), which scores grammaticality on the ordinal scale (from 1 to 4).

(%)	Baseline	E05	E10	E15	E20
0	**91.43**	91.12	90.87	90.61	90.29
5	89.99	**90.00**	89.87	89.72	89.48
10	87.84	87.99	88.07	**88.14**	88.04
15	85.64	86.18	86.54	86.75	**86.82**
20	84.12	84.78	85.28	85.50	**85.76**
∇	-0.37	-0.32	-0.28	-0.26	**-0.23**

Table 4.1: Unlabeled dependency accuracy results with the 5x5 models and test sets. ∇ shows the slope of deterioration in parser performance.

	E05	E10	E15	E20
# edited sents (out of 5,124)	175	391	583	861
grammaticality (source)	2.92	2.95	2.95	2.89
grammaticality (this work)	2.96	2.99	**3.27**	**2.98**

Table 4.2: Grammaticality scores by 1-4 scale regression model (Heilman et al., 2014). The first row shows the number of sentences that are made (at least one) change. Bold numbers show statistically significant improvements.

4.3.2 Results

Figure 4.2 and Table 4.1 shows the result of unlabeled dependency accuracy (UAS).[4] As previously presented (Foster, 2007; Cahill, 2015), our experiment also shows that parser performance deteriorated as the error rate in the test corpus increased. On the error-free test set (0%), the baseline (EF pipeline) outperforms other EREF models; the accuracy is lower when the parser is trained on noisier data. The difference among the models becomes small when the test set has 10% error injection rate. As the rate increases further, the trend of parser accuracy reverses. When the test set has 15% or higher noise, the E20 is the most accurate parser. This trend is presented by the slope of deterioration $\nabla =$

[4]Technically, it is possible to train the model with learning labels simultaneously (LAS), but there is a trade-off between search space and training time. The primary goal of this experiment is to see if the EREF is able to detect and correct grammatical errors.

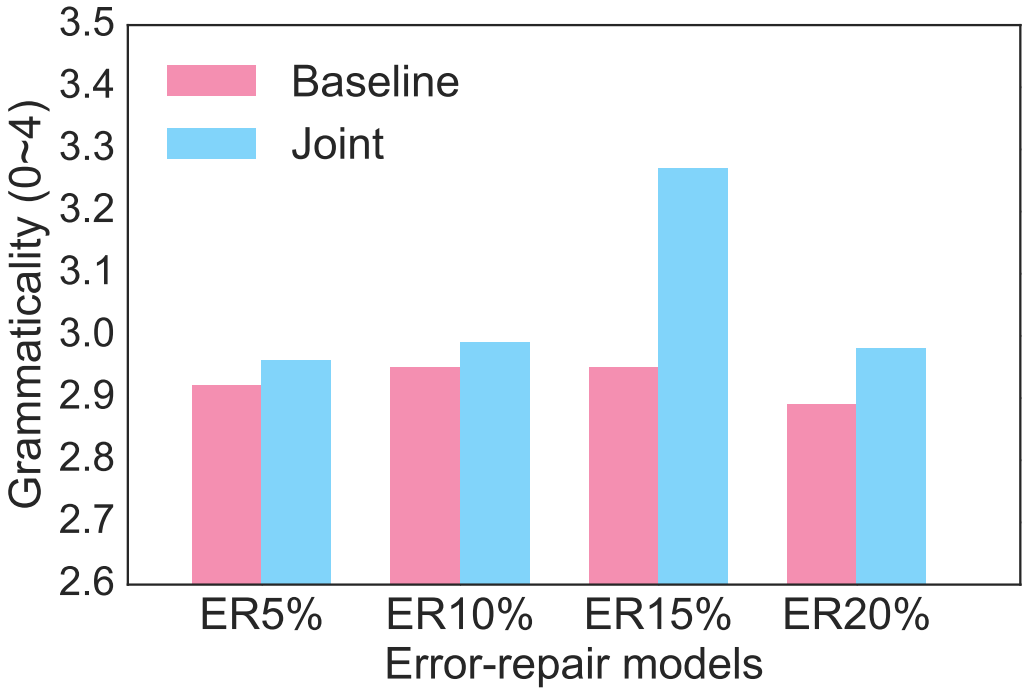

Figure 4.3: Grammaticality scores by 1-4 scale regression model (Heilman et al., 2014).

$\frac{\text{accuracy}_{20\%} - \text{accuracy}_{0\%}}{20}$ in Table 4.1; a parser trained on noisier training data shows smaller decline and more robustness.[5] This indicates that the EREF is more robust than the vanilla EF on ungrammatical texts by jointly parsing and correcting errors.

Figure 4.3 and Table 4.2 demonstrates the result of grammaticality improvement (1-4 scale) on the TLE corpus, and Table 4.3 shows successful and failure corrections by EREF. Minimally trained models (E05 and E10) show little improvement in grammaticality because the models are too conservative to make edits. The models with higher error-injection rates (E15 and E20) achieve 0.1 to 0.3 improvements that are statistically significant. There is still room to improve regarding the amount of corrections. This is probably because TLE

[5]Baseline model *without* preprocessing always underperformed the preprocessed baseline.

Successful cases
I 'm looking forward to [-see-] {+seeing+} you next summer
I 've never [-approve-] {+approved+} his deal
There is not {+a+} possibility to travel
Failure cases
I 've [-assisted-] {+assisting+} your new musical show
I am writing to complain [-about-] {+with+} somethings
I hope you liked {+the+} everything I said

Table 4.3: Successful and failure examples by EREF. The edits are represented by [-deletion-] and {+insertion+}. Adjacent pairs of deletion and insertion are considered as substitution.

contains a variety of errors (e.g., collocation, punctuation) in addition to the five error types we focus. To deal with other error types, we can extend EREF by adding more actions, although it increases the search space.

From a practical perspective, the level of ungrammaticality should be realized ahead of time. This is an issue to be addressed in future research.

4.4 Conclusions

In this chapter, we have presented a joint model that corrects grammatical errors with deriving the dependency tree. The model is an *error-repair* variant of the *non-directional easy-first* dependency parser. We have introduced three new actions, SUBSTITUTE, INSERT, and DELETE into the parser so that it jointly parses and corrects grammatical errors in a sentence. To address the issue of parsing incompletion due to the new actions, we have proposed simple constraints that keep track of editing history for each token and the total number of edits during derivation. The experimental result has demonstrated robustness of

EREF parsers against EF and grammaticality improvement. Our work is positioned at the intersection of noisy text parsing and grammatical error correction. The EREF is a flexible formalism not only for grammatical error correction but other tasks with jointly editing and parsing a given sentence.

In a broader context beyond token-level grammatical error correction, this work lies at the intersection of parsing non-canonical texts and grammatical error correction. Joint dependency parsing and disfluency detection has been pursued (Rasooli and Tetreault, 2013; Rasooli and Tetreault, 2014; Honnibal and Johnson, 2014; Wu et al., 2015; Yoshikawa, Shindo, and Matsumoto, 2016), where a parser jointly parses and detects disfluency (e.g., reparandum and interregnum) for a given speech utterance. Our work could be considered an extension via adding SUBSTITUTE and INSERT actions, although we depend on easy-first non-directional parsing framework instead of a left-to-right strategy. Importantly, the DELETE action is easier to handle than the SUBSTITUTE and INSERT actions, because they bring us challenging issues about a process of candidate word generation and avoiding an infinite loop in derivation. We have addressed these issues as explained in §4.2.2.

In terms of the literature from grammatical error correction, this work is closely related to Dahlmeier and Ng (2012a), where they show an error correction decoder with the easy-first strategy. The decoder iteratively corrects the most probable ungrammatical token by applying different classifiers for each error type. The EREF parser also depends on the easy-first strategy to find ungrammatical index to be deleted, inserted, or substituted, but it parses and corrects errors jointly whereas the decoder is designed as a grammatical error

correction framework rather than a parser.

There is a line of work for parsing ungrammatical sentences (e.g., web forum) by adapting an *existing* parsing scheme on domain specific annotations (Petrov and McDonald, 2012; Cahill, 2015; Berzak et al., 2016; Nagata and Sakaguchi, 2016). Although we share an interest with respect to dealing with ungrammatical sentences, EREF focuses on the parsing scheme for repairing grammatical errors instead of adapting a parser with a domain specific annotation scheme.

More broadly, our work can also be regarded as one of the joint parsing and text normalization tasks such as joint spelling correction and POS tagging (Sakaguchi et al., 2012), word segmentation and POS tagging (Kaji and Kitsuregawa, 2014; Qian et al., 2015).

Now, we have looked at character and token-level error correction models. Before discussing "whole-sentence" error correction models, in the next couple of chapters, we reassess the goal of sentence-level grammatical error correction with respect to the evaluation metrics and benchmarking dataset. We will come back to a whole-sentence error correction model in Chapter 7.

Chapter 5

Reassessing the Goals of Whole Sentence Error Correction[1]

In the previous chapters, we have looked at character-level and token-level error corrections. In this chapter, we move on to sentence-level error correction. In sentence-level error correction, the input is a sequence of tokens, and GEC systems are expected to correct sentence-level grammatical errors in addition to those at the token level.[2]

Before modeling sentence-level GEC systems, in this chapter, we look at some issues in conventional evaluation metrics and datasets, because the evaluation for sentence-level error correction is not as straightforward as character- or token-level error correction. In sentence-level error correction, the input and output can be totally different (e.g., paraphrasing and/or moving phrases); therefore, it might not be appropriate to evaluate sentence-level

[1] Much of this chapter was originally published in Sakaguchi et al. (2016).

[2] Some might wonder what the difference is between phrase- and sentence-level errors. In this thesis, they are treated as the same concept where granularity of errors is longer than or equal to a single token.

GEC systems by accuracy or F-measure-based metrics as in character and token-level error correction.

5.1 Introduction

What is the purpose of grammatical error correction (GEC)? One response is that GEC aims to help people become better writers by correcting grammatical mistakes in their writing. In the NLP community, the original scope of GEC was correcting targeted error types with the goal of providing feedback to non-native writers (Chodorow and Leacock, 2000; Dale and Kilgarriff, 2011; Leacock et al., 2014). As systems improved and more advanced methods were applied to the task, the definition evolved to *whole-sentence correction*, or correcting all errors of every error type (Ng et al., 2014). With this pivot, we urge the community to revisit the original question.

It is often the case that writing exhibits problems that are difficult to ascribe to specific grammatical categories. Consider the following example, repeated from Chapter 1:

Original: From this scope , social media has shorten our distance .

Corrected: From this scope , social media has shortened our distance .

If the goal is to correct verb errors, the grammatical mistake in the original sentence has been addressed and we can move on. However, when we aim to correct the sentence as a whole, a more vexing problem remains. The more prominent error has to do with how unnaturally this sentence reads. The meanings of words and phrases like *scope* and

the corrected *shortened our distance* are clear, but this is not how a native English speaker would use them. A more fluent version of this sentence would be the following:

Fluent: From this perspective , social media has shortened the distance between us .

This issue argues for a broader definition of grammaticality that we will term *native-language fluency*, or simply *fluency*. One can argue that traditional understanding of grammar and grammar correction encompasses the idea of native-language fluency. However, the metrics commonly used in evaluating GEC undermine these arguments. The performance of GEC systems is typically evaluated using metrics that compute corrections against *error-coded* corpora, which impose a taxonomy of types of grammatical errors. Assigning these codes can be difficult, as evidenced by the low agreement found between annotators of these corpora. It is also quite expensive. But most importantly, as we will show in this paper, annotating for explicit error codes places a downward pressure on annotators to find and fix concrete, easily-identifiable grammatical errors (such as *wrong verb tense*) in lieu of addressing the native fluency of the text.

A related problem is the presence of multiple evaluation metrics computed over error-annotated corpora. Recent work has shown that metrics like M^2 and I-measure, both of which require error-coded corpora, produce dramatically different results when used to score system output and produce a ranking of systems in conventional competitions (Felice and Briscoe, 2015).

In light of all of this, we suggest that the GEC task has overlooked a fundamental question: What are the best practices for corpus annotation and system evaluation? This work

attempts to answer this question. We show that native speakers prefer text that exhibits fluent sentences over ones that have only minimal grammatical corrections. We explore different methods for corpus annotation (with and without error codes, written by experts and non-experts) and different evaluation metrics to determine which configuration of annotated corpus and metric has the strongest correlation with the human ranking. In so doing, we establish a reliable and replicable evaluation procedure to help further the advancement of GEC methods.[3] To date, this is the only work to undertake a comprehensive empirical study of annotation *and* evaluation. As we will show, the two areas are intimately related.

In essence, we reframe sentence-level error correction as one of the sentence *generation* tasks such as paraphrasing and machine translation. We propose the most reliable combination of evaluation metric and benchmark dataset by looking at correlations to human judgments. Our experimental results show that the GLEU metric (Napoles et al., 2015) and "fluent" sentence rewrites as the oracle produce strong to very strong correlations with human judgments (Spearman's $\rho = 0.82$, Pearson's $r = 0.73$). We also show that the "fluent" rewrites are simpler and more cost efficient to collect than conventional annotations.

[3]All the scripts and new data we collected are available at `https://github.com/keisks/reassess-gec`.

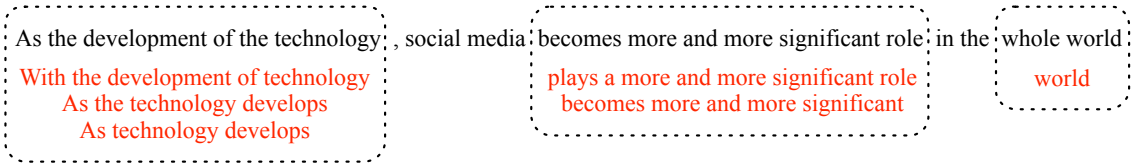

Figure 5.1: An ungrammatical sentence that can be corrected in different ways.

5.2 Current issues in GEC

In this section, we will address issues of the GEC task, reviewing previous work with respect to error annotation and evaluation metrics.

5.2.1 Annotation methodologies

Existing corpora for GEC are annotated for errors using fine-grained coding schemes. To create error-coded corpora, trained annotators must identify spans of text containing an error, assign codes corresponding to the error type, and provide corrections to those spans for each error in the sentence.

One of the main issues with coded annotation schemes is the difficulty of defining the granularity of error types. These sets of error tags are not easily interchangeable between different corpora. Specifically, two major GEC corpora have different taxonomies: the Cambridge Learner Corpus (CLC) (Nicholls, 2003) has 80 tags, which generally represent the word class of the error and the type of error (such as replace preposition, unnecessary pronoun, or missing determiner). In contrast, the NUS Corpus of Learner English (NUCLE) (Dahlmeier, Ng, and Wu, 2013) has only 27 error types. A direct conversion be-

tween them, if possible, would be very complex. Additionally, it is difficult for annotators to agree on error annotations, which complicates the annotation validity as a gold standard (Leacock et al., 2014). This is due to the nature of grammatical error correction, where there can be diverse correct edits for a sentence (Figure 5.1). In other words, there is no single gold-standard correction. The variety of error types and potential correct edits result in very low inter-annotator agreement (IAA), as reported in previous studies (Tetreault and Chodorow, 2008; Rozovskaya and Roth, 2010; Bryant and Ng, 2015).

This leads to a more fundamental question: *why do we depend so much on fine-grained, low-consensus error-type annotations as a gold standard for evaluating GEC systems?*

One answer is that error tags can be informative and useful to provide feedback to language learners, especially for specific closed-class error types (such as determiners and prepositions). Indeed, the CLC, the first large-scale corpus of annotated grammatical errors, was coded specifically with the intent of gathering statistics about errors to inform the development of tools to help English language learners (Nicholls, 2003). Later GEC corpora adhered to the same error-coding template, if not the same error types (Rozovskaya and Roth, 2010; Yannakoudakis, Briscoe, and Medlock, 2011; Dahlmeier, Ng, and Wu, 2013).

The first shared task in GEC aspired to the CLC's same objective: to develop tools for language learners (Dale and Kilgarriff, 2011). Subsequent shared tasks (Dale, Anisimoff, and Narroway, 2012; Ng et al., 2013) followed suit, targeting specific error types. Error-coded corpora are effective training and evaluation data for targeted error correction, and

statistical classifiers have been developed to handle errors involving closed-class words (Rozovskaya and Roth, 2014). However, the 2014 CoNLL shared task engendered a sea change in GEC: in this shared task, systems needed to correct all errors in a sentence, of all error types, including ones more stylistic in nature (Ng et al., 2014). The evaluation metrics and annotated data from the previous shared task were used; however we argue that they do not align with the use case of this reframed task. What is the use case of whole-sentence correction? It should not be to provide specific targeted feedback on error types, but rather to rewrite sentences as a proofreader would.

The community has already begun to view whole-sentence correction as a task, with the yet unstated goal of improving the overall *fluency* of sentences. Independent papers published *human* evaluations of the shared task system output (Napoles et al., 2015; Grundkiewicz, Junczys-Dowmunt, and Gillian, 2015), asking judges to rank systems based on their grammaticality. As GEC moves toward correcting an entire sentence instead of targeted error types, the myriad acceptable edits will result in much lower IAA, compromising evaluation metrics based on the precision and recall of coded errors. At this juncture, it is crucial that we examine whether error-coded corpora and evaluation are necessary for this new direction of GEC.

Finally, it would be remiss not to address the cost and time of corpus annotation. Tetreault and Chodorow (2008) noted that it would take 80 hours to correct 1,000 preposition errors by one trained annotator. Bryant and Ng (2015) reported that it took about three weeks (504 hours) to collect 7 independent annotations for 1,312 sentences, with all

28 CoNLL-2014 error types annotated. Clearly, constructing a corpus with fine-grained error annotations is a labor-intensive process. Due to the time and cost of annotation, the corpora currently used in the community are few and tend to be small, hampering robust evaluations as well as limiting the power of statistical models for generating corrections. If an effective method could be devised to decrease time or cost, larger corpora—and more of them—could be created. There has been some work exploring this, namely Tetreault and Chodorow (2008), which used a sampling approach that would only work for errors involving closed-class words. Pavlick et al. (Pavlick, Yan, and Callison-Burch, 2014) also describe preliminary work into designing an improved crowdsourcing interface to expedite data collection of coded errors.

Section 5.3 outlines our annotation approach, which is faster and cheaper than previous approaches because it does not make use of error coding.

5.2.2 Evaluation practices

As briefly introduced in Chapter 2, three GEC specific evaluation metrics have been proposed for GEC in addition to F-score: MaxMatch (M^2) (Dahlmeier and Ng, 2012b), I-measure (Felice and Briscoe, 2015), and GLEU (Napoles et al., 2015). The first two compare the changes made in the output to error-coded spans of the reference corrections. M^2 was the metric used for the 2013 and 2014 CoNLL GEC shared tasks (Ng et al., 2013; Ng et al., 2014). It captures word- and phrase-level edits by building an edit lattice and calculating an F-score over the lattice.

Felice and Briscoe (2015) note problems with M^2: specifically, it does not distinguish between a "do-nothing baseline" and systems that only propose wrong corrections; also, phrase-level edits can be easily gamed because the lattice treats the deletion of a long phrase as a single edit. To address these issues, they propose I-measure, which generates a token-level alignment between the source sentence, system output, and gold-standard sentences, and then computes accuracy based on the alignment.

Unlike these approaches, GLEU does not use error-coded references[4] (Napoles et al., 2015). Based on BLEU (Papineni et al., 2002), it computes *n*-gram precision of the system output against reference sentences. GLEU additionally penalizes text in the output that was unchanged from the source but changed in the reference sentences.

Recent work by Napoles et al. (Napoles et al., 2015) and Grundkiewicz et al. (Grundkiewicz, Junczys-Dowmunt, and Gillian, 2015) evaluated these metrics against human evaluations obtained using methods borrowed from the Workshop on Statistical Machine Translation (Bojar et al., 2014). Both papers found a moderate to strong correlation with human judgments for GLEU and M^2, and a slightly negative correlation for I-measure. Importantly, however, none of these metrics achieved as a high correlation with the human oracle ranking as desired in a fully reliable metric.

In §5.4, we examine the available metrics over different types of reference sets to identify an evaluation setup nearly as reliable as human experts.

[4]We use the term *references* to refer to the corrected sentences, since the term *gold standard* suggests that there is just one right correction.

	Technically grammatical	**Not technically grammatical**
Fluent	In addition, it is impractical to make such a law.	I don't like this book, it's really boring.
Not fluent	Firstly , someone having any kind of disease belongs to his or her privacy .	It is unfair to release a law only point to the genetic disorder.

Table 5.1: Examples and counterexamples of technically grammatical and fluent sentences.

Original	Genetic disorder may or may not be hirataged hereditary disease and it is sometimes hard to find out one has these kinds of diseases .
Expert fluency	A genetic disorder may or may not be □ a hereditary disease , and it is sometimes hard to find out whether one has these kinds of diseases .
Non-expert fluency	Genetic □ factors can manifest overtly as disease □ , or simply be carried , making it □ hard , sometimes , to find out if one has □ a genetic predisposition to disease .

Table 5.2: An example sentence with expert and non-expert fluency edits. Moved and changed or inserted spans are underlined and □ indicates deletions.

5.3 Creating a new, *fluent* GEC corpus

We hypothesize that human judges, when presented with two versions of a sentence, will favor *fluent* versions over ones that exhibit only *technical grammaticality*.

By *technical grammaticality*, we mean adherence to an accepted set of grammatical conventions. In contrast, we consider a text to be *fluent* when it looks and sounds natural to a native-speaking population. Both of these terms are hard to define precisely, and fluency especially is a nuanced concept for which there is no checklist of criteria to be met.[5] To carry the intuitions, Table 5.1 contains examples of sentences that are one, both, or neither.

[5]It is important to note that both grammaticality and fluency are determined with respect to a particular speaker population and setting. In this paper, we focus on Standard Written English, which is the standard used in education, business, and journalism. While judgments of individual sentences would differ for other populations and settings (for example, spoken African-American Vernacular English), the distinction between grammaticality and fluency would remain.

A text does not have to be technically grammatical to be considered fluent, although in almost all cases, fluent texts are also technically grammatical. In the rest of this paper, we will demonstrate how they are quantifiably different with respect to GEC.

Annotating coded errors encourages a minimal set of edits because more substantial edits often address overlapping and interacting errors. For example, the annotators of the NUCLE corpus, which was used for the recent shared tasks, were explicitly instructed to select the minimal text span of possible alternatives (Dahlmeier, Ng, and Wu, 2013). There are situations where error-coded annotations are useful to help students correct specific grammatical errors. The ability to do this with the non-error-coded, fluent annotations we advocate here is no longer direct, but is not lost entirely. For this purpose, some recent studies have proposed *post hoc* automated error-type classification methods (Swanson and Yamangil, 2012; Xue and Hwa, 2014), which compare the original sentence to its correction and deduce the error types.

We speculate that, by removing the error-coding restraint, we can obtain edits that sound more fluent to native speakers while also reducing the expense of annotation, with diminished time and training requirements. Chodorow et al. (2012) and Tetreault, Chodorow, and Madnani (2014) suggested that it is better to have a large number of annotators to reduce bias in automatic evaluation. Following this recommendation, we collected additional annotations without error codes, written by both experts and non-experts.

5.3.1 Data collection

We collected a large set of additional human corrections to the NUCLE 3.2 test set[6], which was used in the 2014 CoNLL Shared Task on GEC (Ng et al., 2014) and contains 1,312 sentences error-coded by two trained annotators. Bryant and Ng (2015) collected an additional eight annotations using the same error-coding framework, referred to here as BN15.

We collected annotations from both experts and non-experts. The experts[7] were three native English speakers familiar with the task. To ensure that the edits were clean and meaning-preserving, each expert's corrections were inspected by a different expert in a second pass. For non-experts, we used crowdsourcing, which has shown potential for annotating closed-class errors as effectively as experts (Tetreault, Filatova, and Chodorow, 2010; Madnani et al., 2011; Tetreault, Chodorow, and Madnani, 2014). We hired 14 participants on Amazon Mechanical Turk (MTurk) who had a HIT approval rate of at least 95% and were located in the United States. The non-experts went through an additional screening process: before completing the task, they wrote corrections for five sample sentences, which were checked by the three experts.[8]

We collected four complete sets of annotations by both types of annotators: two sets of *minimal edits*, designed to make the original sentences *technically grammatical* (following the NUCLE annotation instructions but without error coding), and two sets of *fluency edits*,

[6] `www.comp.nus.edu.sg/~nlp/conll14st.html`
[7] All of the expert annotators are authors of Sakaguchi et al. (2016).
[8] The experts verified that the participants were following the instructions and not gaming the HITs.

designed to elicit native-sounding, *fluent* text. The instructions were:

- *Minimal edits:* Make the smallest number of changes so that each sentence is grammatical.
- *Fluency edits:* Make whatever changes necessary for sentences to appear as if they had been written by a native speaker.

In total, we collected 8 ($2 \times 2 \times 2$) annotations from each original sentence (minimal and fluency, expert and non-expert, two corrections each). Of the original 1,312 sentences, the experts flagged 34 sentences that needed to be merged together, so we skipped these sentences in our analysis and experiments. In the next subsection, we show how humans rate edits made by experts and non experts in both minimal and fluency edits.

5.3.2 Human evaluation

As an additional validation, we ran a task to establish the relative quality of the new fluency and minimal-edit annotations using crowdsourcing via Amazon Mechanical Turk. Participants needed to be in the United States with a HIT approval rate of at least 95% and pass a preliminary ranking task, graded by the authors. We randomly selected 300 sentences and asked participants to rank the new annotations, one randomly selected NUCLE correction, and the original sentence in order of grammaticality and meaning preservation (that is, a sentence that is well-formed but changes the meaning of the original source should have a lower rank than one that is equally well-formed but maintains the original

Original	Some family may feel hurt , with regards to their family pride or reputation , on having the knowledge of such genetic disorder running in their family .
NUCLE	Some family members may feel hurt ▯ with regards to their family pride or reputation ▯ on having the knowledge of a genetic disorder running in their family .
Expert fluency	On ▯ learning of such a genetic disorder running in their family , some family members may feel hurt ▯ regarding their family pride or reputation .
Non-expert fluency	Some relatives may ▯ be concerned about the family 's ▯ reputation – not to mention their own pride – in relation to this news of ▯ familial genetic defectiveness ▯ .
Expert minimal	Some families may feel hurt ▯ with regards to their family pride or reputation , on having ▯ knowledge of such a genetic disorder running in their family .
Non-expert minimal	Some family may feel hurt ▯ with regards to their family pride or reputation ▯ on having the knowledge of such genetic disorder running in their family .

Table 5.3: An example sentence with the original NUCLE correction and fluency and minimal edits written by experts and non-experts. Moved and changed or inserted spans are underlined and ▯ indicates deletions.

meaning). Since we were comparing the minimal edits to the fluency edits, we did not define the term *grammaticality*, but instead relied on the participants' understanding of the term. Each sentence was ranked by two different judges, for a total of 600 rankings, yielding 7,795 pairwise comparisons.

To rank systems, we use the TrueSkill approach (Herbrich, Minka, and Graepel, 2006; Sakaguchi, Post, and Van Durme, 2014), based on a protocol established by the Workshop on Machine Translation (Bojar et al., 2014; Bojar et al., 2015). For each competing system, TrueSkill infers the absolute system quality from the pairwise comparisons, representing each as the mean of a Gaussian. These means can then be sorted to rank systems. By running TrueSkill 1,000 times using bootstrap resampling and producing a system ranking

#	Score	Range	Annotation type
1	1.164	1–2	Expert fluency
	0.976	1–2	Non-expert fluency
3	0.540	3	NUCLE
4	0.265	4	Expert minimal
5	-0.020	5	Non-expert minimal
6	-2.925	6	Original sentence

Table 5.4: Human ranking of the new annotations by grammaticality. Lines between systems indicate clusters according to bootstrap resampling at $p \leq 0.05$. Systems in the same cluster are considered to be tied.

each time, we collect a range of ranks for each system. We can then cluster systems according to non-overlapping rank ranges (Koehn, 2012) to produce the final ranking, allowing ties.

Table 5.4 shows the ranking of "grammatical" judgments for the additional annotations and the original NUCLE annotations. While the score of the expert fluency edits is higher than the non-expert fluency, they are within the same cluster, suggesting that the judges perceived them to be just as good. The fluency rewrites by both experts and non-experts are clearly preferable over the minimal edit corrections, although the error-coded NUCLE corrections are perceived as more grammatical than the minimal corrections.

5.4 What is the Best Annotation–Evaluation Combination?

We have shown that humans prefer fluency edits to error-coded and minimal-edit corrections, but it is not clear whether these annotations are still an effective reference for automatic evaluation metrics. In particular, the most changes made by fluency edits can make it more challenging for us to use the automated evaluation metrics. In this section, we investigate the impact of the combination difference between (1) reference sets and (2) automated evaluation metrics for sentence-level GEC. Namely, with reference sets having such different characteristics, the natural question is: which reference and evaluation metric pairing best reflects human judgments of grammaticality?

To answer this question, we performed an exhaustive comparisons of existing metrics and annotation sets by evaluating the 12 system outputs made public from the 2014 CoNLL Shared Task. To our best of knowledge, this is the first attempt that the interplay of annotation scheme and evaluation metric, as well as the rater expertise, has been evaluated jointly for GEC.[9]

[9]This section is primarily contributed by Courtney Napoles. Please refer her thesis for more details.

5.4.1 Experiments

We investigate four automated metrics: BLEU, M^2, I-measure,[10] and GLEU. BLEU itself is not designed for GEC but we include it in this experiment, because evaluation of sentence-level GEC is similar to that of machine-translation, which considers overlap instead of absolute alignment between the output and reference sentences. As introduced in Chapter 2 (§ 2.1), GLEU is a variant of BLEU that adds an additional penalty for tokens that should have been edited but weren't. For the M^2 and I-measure metrics, we aligned the source and hypothesis/reference using dynamic programming (i.e., Levenshtein edit distance algorighm).[11]

With each metric, we compare the system outputs to each of the six annotation sets. We add one more annotation set *all*, by combining all the six annotations. The systems are ranked based on their scores by the cross-product of each metric–annotation-set pair, and thus generated a total of 28 different rankings (4 metrics $\times$ 7 annotation sets).

To find out the best metric, we compared the system ranking obtained from each evaluation technique against the expert human ranking reported in Grundkiewicz, Junczys-Dowmunt, and Gillian (2015) (Table 5.5).[12]

[10] I-measure is conducted with the `-nomix` flag, preventing the algorithm from finding the optimal alignment across all possible edits. Alignment was very memory-intensive and time consuming, even when skipping long sentences.

[11] Costs for insertion, deletion, and substitution are set to 1, allowing partial match (e.g., same lemma). Neither metric makes use of the annotation labels, so we simply assigned dummy error codes.

[12] We could use the expert human ranking reported in Napoles et al. (2015), but the ranking is obtained from randomly sampled sentences in the test set, while Grundkiewicz, Junczys-Dowmunt, and Gillian (2015) use the entire test set. In both work, the human ranking is obtained by TrueSkill algorithm (Herbrich, Minka, and Graepel, 2006; Sakaguchi, Post, and Van Durme, 2014). The technical details are explained in Appendix A.

Rank	Score	Range	System
1	0.273	1	AMU
2	0.182	2	CAMB
3	0.114	3-4	RAC
	0.105	3-5	CUUI
	0.080	4-5	POST
4	-0.001	6-7	PKU
	-0.022	6-8	UMC
	-0.041	7-10	UFC
	-0.055	8-11	IITB
	-0.062	8-11	input
	-0.074	9-11	SJTU
5	-0.142	12	NTHU
6	-0.358	13	IPN

Table 5.5: System ranking obtained from human ranking (adapted from Grundkiewicz, Junczys-Dowmunt, and Gillian (2015).

5.4.2 Results

Table 5.6 shows the correlation of the expert rankings with all of the evaluation configurations. Among all the combinations, the GLEU metric with expert fluency has the highest correlation with human judgements. In other words, this is the best configuration for evaluating GEC systems at this moment.

M^2 and GLEU, and the expert fluency annotations had stronger positive correlations than the non-expert annotations. It is important that just two expert fluency annotations with GLEU have the strongest correlation with the human ranking out of all other metric–reference combinations ($\rho = 0.819$, $r = 0.731$), and it is more efficient with respect to the annotation cost and time than the error-coded metrics such as M^2 and I-measure. Expert fluency with M^2 is the third-best pairing. It is interesting to see that M^2 has the strongest correlations with Expert-minimal ($\rho = 0.775$) annotation on NUCLE ($r = 0.677$), which

	M^2	GLEU	I-measure	BLEU
NUCLE	0.725	0.626	-0.423	-0.456
	0.677*	0.646	-0.313	-0.310
BN15	0.692	0.720	-0.066	-0.319*
	0.641	0.697	-0.007	-0.255
Expert fluency	0.758	**0.819***	-0.297	-0.385
	0.665	**0.731***	-0.256	-0.230*
Non-expert fluency	0.703	0.676	-0.451	-0.451
	0.655	0.668	-0.319	-0.388
Expert minimal	0.775*	0.786	-0.467	-0.456
	0.655	0.676	-0.385	-0.396
Non-expert minimal	0.769	-0.187	-0.467	-0.495
	0.641	-0.110	-0.402	-0.473
All	0.692	0.725	-0.055*	-0.462
	0.617	0.724	0.061*	-0.314

Table 5.6: Correlation between the human ranking and metric scores over different reference sets. The first line of each cell is Spearman's ρ and the second line is Pearson's r. The strongest correlations for each metric are starred, and the overall strongest correlations are in bold.

may support that M^2 is designed especially for minimal edits. Furthermore, it is an important finding that the non-expert fluency edits reasonably correlate with both M^2 and GLEU, which indicates the possibility of more cost efficient annotations by crowdsourcing.

A larger number of references could improve performance for GLEU. Because fluency edits tend to have more variations than error-coded minimal-edit annotations, it is not obvious how many fluency edits are necessary to cover the full range of possible corrections. To address this question, we ran an additional small-scale experiment, where we collected 10 non-expert fluency edits for 20 sentences and computed the average GLEU scores of the submitted systems against an increasing number of these fluency references. The result (Figure 5.2) shows that the GLEU score with more fluency references, but the effect

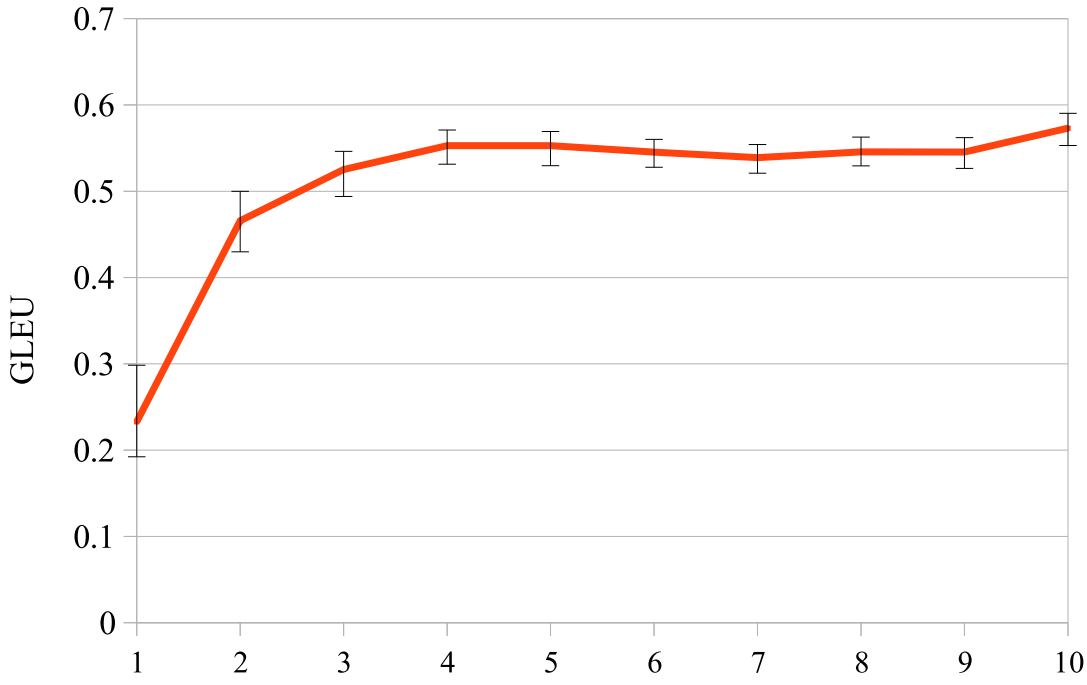

Figure 5.2: Mean GLEU scores with different numbers of fluency references. The red line corresponds to the average GLEU score of the 12 GEC systems and the vertical bars show the maximum and minimum GLEU scores.

starts to level off when there are at least 4 references, suggesting that 4 references cover the majority of possible changes. A similar pattern was observed by Bryant and Ng (2015) in error-coded annotations with the M^2 metric.

As Grundkiewicz, Junczys-Dowmunt, and Gillian (2015) and Napoles et al. (2015) have shown, I-measure and BLEU are shown to be unfavorable evaluation metrics for GEC. Although BLEU and GLEU both depend on the *n*-gram matching overlap between the hypothesis and original sentences, GLEU shows strong positive correlations with human rankings whereas BLEU has a moderate negative correlation. As explained in Chapter 2 (§2.1), the advantage of GLEU comes from penalizing *n*-gram in the hypothesis H (i.e., system out-

put) that were present in the source (S) and absent from the reference (R). Mathematically, the difference between GLEU and BLEU is how to compute the precision p'_n:

$$\begin{aligned} \text{BLEU} : p'_n &= \frac{N(H,R)}{N(H)} \\ \text{GLEU} : p'_n &= \frac{N(H,R) - [N(H,S) - N(H,S,R)]}{N(H)}. \end{aligned} \tag{5.1}$$

In short, in GLEU, a system is penalized by missing *n*-grams that should have been corrected (or at least edited), whereas BLEU has no such penalty term and instead only rewards *n*-grams that occur in the references and the hypothesis. This becomes problematic in monolingual text rewriting tasks where there is significant overlap between the reference and the original sentences, because conservative (e.g., minimal editing) models are more benefited unfairly than radical (e.g., fluency editing) models. Let's take a look at the following example.

Source:	*I looks forward to see you.
Reference:	I look forward to seeing you.
Hypothesis 1:	I looks forward to see you.
Hypothesis 2:	I looked forward to meeting you.

The first hypothesis is conservative (i.e., no change), and the second one is radically edited. In BLEU metric (in the case of $n = 1$), both hypotheses have the same precision (i.e, 4 out of 6 tokens are overlapped with the reference), and there is no way to make

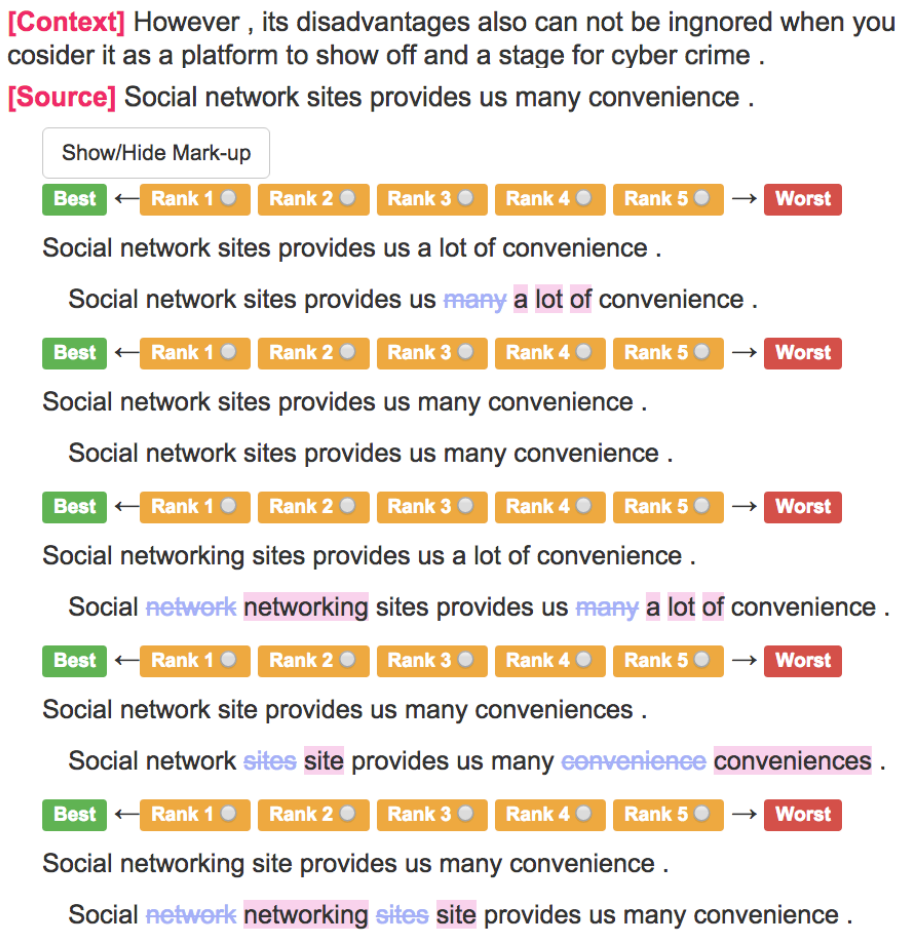

Figure 5.3: Screenshot of the GEC grammaticality judgment task (HIT).

distinction between "do-nothing" and "aggressive edits". GLEU addresses this issue by discouraging the "do-nothing" strategy (i.e., two tokens, *looks* and *see*, are penalized).

5.5 GEC System Evaluation by Non-experts

Automatic metrics are only a proxy for human judgments, which are crucial to truthfully ascertain the quality of systems. Even the best result in §5.4.2, which is state of the art

and has very strong rank correlation ($\rho = 0.819$) with the expert ranking, makes dramatic errors in the system ranking. Given the inherent imperfection of automatic evaluation (and possible over-optimization to the NUCLE data set), we recommend that human evaluation be produced alongside metric scores whenever possible. However, human judgments can be expensive to obtain. Crowdsourcing may address this problem and has been shown to yield reasonably good judgments for several error types at a relatively low cost (Tetreault, Chodorow, and Madnani, 2014). Therefore, we apply crowdsourcing to sentence-level grammaticality judgments, by replicating previous experiments that reported expert rankings of system output (Napoles et al., 2015; Grundkiewicz, Junczys-Dowmunt, and Gillian, 2015) using non-experts on MTurk.

5.5.1 Experiments

Using the same data set as those experiments and the work described in this paper, we asked screened participants[13] on MTurk to rank five randomly selected systems and NUCLE corrections from best to worst, with ties allowed. 294 sentences were randomly selected for evaluation from the NUCLE subsection used in Grundkiewicz, Junczys-Dowmunt, and Gillian (2015), and the output for each sentence was ranked by two different participants. The 588 system rankings yield 26,265 pairwise judgments, from which we inferred the absolute system ranking using TrueSkill.

[13]Participants in the United States with a HIT approval rate $\geq$ 95% had to pass a sample ranking task graded by the authors.

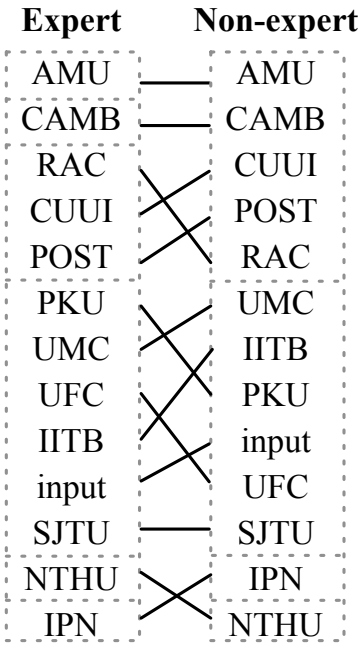

Figure 5.4: Output of system rankings by experts and non-experts, from best to worst. Dotted lines indicate clusters according to bootstrap resampling ($p \leq 0.05$).

Judges	κ	κ_w
Non-experts	0.29	0.43
Experts	0.29	0.45
Non-experts and Experts	0.15	0.23

Table 5.7: Inter-annotator agreement of pairwise system judgments within non-experts, experts and between them. We show Cohen's κ and quadratic-weighted κ.[15]

5.5.2 Results

Figure 5.4 compares the system ranking by non-experts to the same expert ranking used in § 5.4.1. The rankings have very strong correlation ($\rho = 0.917$, $r = 0.876$), indicating that non-expert grammaticality judgments are comparably as reliable as those by experts. The non-expert correlation can be seen as an upper bound for the task, which is approached but not yet attained by automatic metrics.

Systems in the same cluster, indicated by dotted lines in Figure 5.4, can be viewed as ties. From this perspective the expert and non-expert rankings are virtually identical. In addition, experts and non-experts have similar inter-annotator agreement in their pairwise system judgments (Table 5.7). The agreement between experts and non-experts is lower than the agreement between just experts or just non-experts, which may be due to the difference of these experimental settings for experts (Grundkiewicz, Junczys-Dowmunt, and Gillian, 2015) and for non-experts (this work). However, this finding is not overly concerning since the correlation between the rankings is so strong.

In all, judgments cost approximately $140 ($0.2 per sentence) and took a total of 32 hours to complete. Because the non-expert ranking very strongly correlates to the expert ranking and non-experts have similar IAA as experts, we conclude that expensive expert judgments can be replaced by non-experts, when those annotators have been appropriately screened.

5.6 Conclusion

In this chapter, we attempt to determine the most reliable evaluation framework for GEC systems with respect to the metrics and annotation methodology. What we have found is that there is a real distinction between technical grammaticality and fluency. Fluency is a level of mastery that goes beyond knowledge of how to follow the rules, and includes

[15] In addition to Cohen's κ, we report weighted κ because $A > B$ and $A < B$ should have less agreement than $A > B$ and $A = B$.

knowing when they can be broken or flouted. Language learners—who are a prime constituency motivating the GEC task—ultimately care about the latter. But crucially, the current approach of collecting error-coded annotations places downward pressure on annotators to minimize edits in order to neatly label them. This results in annotations that are less fluent, and therefore less useful, than they should be. We have demonstrated this with the collection of both minimally-edited and fluent rewrites of a common test set (§5.3.1); the preference for fluent rewrites over minimal edits is clear (Table 5.4).

To correct this, the annotations and associated metrics used to score automated GEC systems should be brought more in line with this broadened goal. We advocate for the collection of fluent sentence-level rewrites of ungrammatical sentences, which is cheaper than error-coded annotations and provides annotators with the freedom to produce fluent edits. In the realm of automatic metrics, we found that a modified form of GLEU computed against expert fluency rewrites correlates best with a human ranking of the systems; a close runner-up collects the rewrites from non-experts instead of experts.

Finally, to stimulate metric development, we found that we were able to produce a new human ranking of systems using non-expert judges. These judges produced a ranking that was highly correlated with the expert ranking produced in earlier work (Grundkiewicz, Junczys-Dowmunt, and Gillian, 2015). The implication is further reduced costs in producing a gold-standard ranking for new sets of system outputs against both existing and new corpora.

As a result, we make the following recommendations:

- GEC should be evaluated against 2–4 whole-sentence rewrites, which can be obtained by non-experts.
- Automatic metrics that rely on error coding are not necessary, depending on the use case. Of the automatic metrics that have been proposed, we found that a modified form of GLEU (Napoles et al., 2015) is the best-correlated.
- The field of GEC is in danger from over-reliance on a single annotated corpus (NUCLE). New corpora should be produced in a regular fashion, similar to the Workshop on Statistical Machine Translation.

Fortunately, collecting annotations in the form of unannotated sentence-level rewrites is much cheaper than error-coding, facilitating these practices.

By framing grammatical error correction as fluency, we can reduce the cost of annotation while creating a more reliable gold standard. We have clearly laid improved practices for annotation and evaluation, demonstrating that better quality results can be achieved for less cost using fluency edits instead of error coding. All of the source code and data, including templates for data collection, will be publicly available, which we believe is crucial for supporting the improvement of GEC in the long term.

Following these recommendations, in the next chapter, we develop a new corpus, JHU FLuency-Extended GUG corpus (JFLEG). In order to avoid the over-reliance on a single benchmarking (i.e., NUCLE), the sentences in JFLEG have more diversity than NUCLE corpus, in terms of the topics, and writers' proficiency and native language.

Chapter 6

A Fluency Corpus and Benchmark for Grammatical Error Correction[1]

In the previous chapter, we have investigated the most reliable evaluation framework for GEC systems regarding the metrics and annotation scheme. We have found that (1) fluency edits are more reliable than minimal edits, and (2) fluency edits can be made by non-experts (i.e., crowdsourcing). In this chapter, we build a fluency-oriented GEC corpus that has more diversity in terms of the learners' proficiency and variety of error types in token- and sentence-levels that are not easily categorized by error codes.

[1]Much of this chapter was originally published in Napoles, Sakaguchi, and Tetreault (2017).

6.1 Introduction

As pointed out in the previous chapter, automatic grammatical error correction (GEC) progress is limited by the corpora available for developing and evaluating systems. Following the release of the test set of the CoNLL–2014 Shared Task on GEC (Ng et al., 2014), systems have been compared and new evaluation techniques proposed on this single dataset. This corpus has enabled substantial advancement in GEC beyond the shared tasks, but we are concerned that the field is over-developing on this dataset. This is problematic for two reasons: 1) it represents one specific population of language learners; and 2) the corpus only contains *minimal edits*, which correct the grammaticality of a sentence but do not necessarily make it *fluent* or native-sounding.

To illustrate the need for fluency edits, consider the example in Table 6.1. The correction with only minimal edits is grammatical but sounds *awkward* (unnatural to native speakers). The fluency correction has more extensive changes beyond addressing grammaticality, and the resulting sentence sounds more natural and its intended meaning is more clear. It is not unrealistic to expect these changes from automatic GEC: the current best systems use machine translation (MT) and are therefore capable of making broader sentential rewrites but, until now, there has not been a gold standard against which they could be evaluated.

Following the recommendations in the previous chapter, we release a new corpus for GEC, the **J**HU **FL**uency-**E**xtended **G**UG corpus (JFLEG), which adds a layer of annotation to the GUG corpus (Heilman et al., 2014). GUG represents a cross-section of ungrammati-

Original:	they just creat impression such well that people are drag to buy it .
Minimal edit:	They just create an impression so well that people are dragged to buy it .
Fluency edit:	They just create such a good impression that people are compelled to buy it.

Table 6.1: A sentence corrected with just minimal edits compared to fluency edits.

cal data, containing sentences written by English language learners with different L1s and proficiency levels. For each of 1,511 GUG sentences, we have collected four human-written corrections which contain holistic fluency rewrites instead of just minimal edits. This corpus represents the diversity of edits that GEC needs to handle and sets a gold standard to which the field should aim. We overview the current state of GEC by evaluating the performance of four leading systems on this new dataset. We analyze the edits made in JFLEG and summarize which types of changes the systems successfully make, and which they need to address. JFLEG will enable the field to move beyond minimal error corrections.

6.2 The JFLEG corpus

Our goal in this chapter is to create a new corpus of fluency edits, as shown in the previous chapter, in which we have identified the shortfalls of minimal edits: they artificially restrict the types of changes that can be made to a sentence and do not reflect the types of changes required for native speakers to find sentences *fluent*, or natural sounding. We collected annotations on a public corpus of ungrammatical text, the GUG (Grammatical/Ungrammatical) corpus (Heilman et al., 2014). GUG contains $3.1k$ sentences written by English language learners for the TOEFL® exam, covering a range of topics. The original

GUG corpus is annotated with grammaticality judgments for each sentence, ranging from 1–4, where 4 is perfect or native sounding, and 1 incomprehensible. The sentences were coded by five crowdsourced workers and one expert. We refer to the mean grammaticality judgment of each sentence from the original corpus as the GUG score.

In our extension, JFLEG, the 1,511 sentences which comprise the GUG development and test sets were corrected four times each on Amazon Mechanical Turk. Annotation instructions are included in Table 6.2. 50 participants from the United States passed a qualifying task of correcting five sentences, which was reviewed by the authors (two native and one proficient non-native speakers of American English). Annotators also rated how difficult it was for them to correct each sentence on a 5-level Likert scale (5 being very easy and 1 very difficult). On average, the sentences were relatively facile to correct (mean difficulty of 3.5 ± 1.3), which moderately correlates with the GUG score (Pearson's $r = 0.47$), indicating that less grammatical sentences were generally more difficult to correct. To create a blind test set for the community, we withhold half (747) of the sentences from the analysis and evaluation herein.[2]

6.3 Evaluation

To assess the current state of GEC, we collected automated corrections of JFLEG from four leading GEC systems with no modifications. They take different approaches but all use some form of MT. The best system from the CoNLL-2014 Shared Task is a hybrid ap-

[2]For more detailed analyses, please see the Thesis by Courtney Napoles.

Please correct the following sentence to make it sound natural and fluent to a native speaker of (American) English. The sentence is written by a second language learner of English. You should fix grammatical mistakes, awkward phrases, spelling errors, etc. following standard written usage conventions, but your edits must be conservative. Please keep the original sentence (words, phrases, and structure) as much as possible. The ultimate goal of this task is to make the given sentence sound natural to native speakers of English without making unnecessary changes. Please do not split the original sentence into two or more. Edits are not required when the sentence is already grammatical and sounds natural.

Table 6.2: JFLEG annotation instructions.

proach, combining a rule-based system with MT and language-model reranking (CAMB14; Felice et al., 2014). Other systems have been released since then and report improvements on the 2014 Shared Task. They include a neural MT model (CAMB16; Yuan and Briscoe, 2016), a phrase-based MT (PBMT) with sparse features (AMU; Junczys-Dowmunt and Grundkiewicz, 2016), and a hybrid system that incorporates a neural-net adaptation model into PBMT (NUS; Chollampatt et al., 2016).

We evaluate system output against the four sets of JFLEG corrections with GLEU, an automatic fluency metric specifically designed for this task (Napoles et al., 2015) and the Max-Match metric (M^2) (Dahlmeier and Ng, 2012b). GLEU is based on the MT metric BLEU, and represents the n-gram overlap of the output with the human-corrected sentences, penalizing n-grams that were been changed in the human corrections but left unchanged by a system. It was developed to score fluency in addition to minimal edits since it does not require an alignment between the original and corrected sentences. M^2 was designed to score minimal edits and was used in the CoNLL 2013 and 2014 shared tasks on GEC (Ng

System	TrueSkill	GLEU	M^2	Sentence changed
Original	-1.64	38.2	0.0	–
CAMB16	**0.21**	**47.2**	50.8	74%
NUS	-0.20*	46.3	**52.7**	69%
AMU	-0.46*	41.7	43.2	56%
CAMB14	-0.51*	42.8	46.6	58%
Human	2.60	55.3	63.2	86%

Table 6.3: Scores of system outputs. * indicates no significant difference from each other.

et al., 2013; Ng et al., 2014). Its score is the $F_{0.5}$ measure of word and phrase-level changes calculated over a lattice of changes made between the aligned original and corrected sentences. Since both GLEU and M^2 have only been evaluated on the CoNLL-2014 test set, we additionally collected human rankings of the outputs to determine whether human judgments of relative grammaticality agree with the metric scores when the reference sentences have fluency edits.

The two native English-speaking authors ranked six versions of each of 150 JFLEG sentences: the four system outputs, one randomly selected human correction, and the original sentence. The absolute human ranking of systems was inferred using TrueSkill, which computes a relative score from pairwise comparisons, and we cluster systems with overlapping ranges into equivalence classes by bootstrap resampling (Sakaguchi, Post, and Van Durme, 2014; Herbrich, Minka, and Graepel, 2006). The two best ranked systems judged by humans correspond to the two best GLEU systems, but GLEU switches the order of the bottom two. The M^2 ranking does not perform as well, reversing the order of the top two systems and the bottom two (Table 6.3).[3] The upper bound is GLEU = 55.3 and M^2 = 63.2,

[3]No conclusive recommendation about the best-suited metric for evaluating fluency corrections can be drawn from these results. With only four systems, there is no significant difference between the metric rankings, and even the human rank has no significant difference between three systems.

the mean metric scores of each human correction compared to the other three. CAMB16 and NUS are halfway to the gold-standard performance measured by GLEU and, according to M^2, they achieve approximately 80% of the human performance. The neural methods (CAMB16 and NUS) are substantially better than the other two according to both metrics. This ranking is in contrast to the ranking of systems on the CoNLL-14 shared task test against minimally edited references. On these sentences, AMU, which was tuned to M^2, achieves the highest M^2 score with 49.5 and CAMB16, which was the best on the fluency corpus, ranks third with 39.9.

6.4 Conclusions

This chapter have presented JFLEG, a new corpus for developing and evaluating GEC systems with respect to fluency as well as grammaticality.[4] Our hope is that this corpus will serve as a starting point for advancing GEC beyond minimal error corrections. We described qualitative and quantitative analysis of JFLEG, and benchmarked four leading systems on this data. The relative performance of these systems varies considerably when evaluated on a fluency corpus compared to a minimal-edit corpus, underlining the need for a new dataset for evaluating GEC. Overall, current systems can successfully correct closed-class targets such as number agreement and prepositions errors (with incomplete coverage), but ignore many spelling mistakes and long-range context-dependent errors. Neural methods are better than other systems at making fluency edits, but this may be

[4] https://github.com/keisks/jfleg

at the expense of maintaining the meaning of the input. As there is still a long way to go in approaching the performance of a human proofreader, these results and benchmark analyses help identify specific issues that GEC systems can improve in future research.

Chapter 7

Sentence-level Error Correction: Neural Reinforcement Learning for sentence level GEC[1]

In the previous chapters (5 and 6), we have obtained the most reliable (i.e., correlating with human judgments) evaluation framework for "whole-sentence" GEC regarding the metric, annotation, and dataset. Finally, in this chapter, we propose a sentence-level GEC model that is directly optimized toward the reassessed goal of GEC, namely *fluency*.

[1]Much of this chapter was originally published in Sakaguchi, Post, and Van Durme (2017b).

7.1 Introduction

Research in automated Grammatical Error Correction (GEC) has expanded from token-level, closed class corrections (e.g., determiners, prepositions, verb forms) to phrase-level, open class issues that consider fluency (e.g., content word choice, idiomatic collocation, word order, etc.).

The expanded goals of GEC have led to new proposed models deriving from techniques in data-driven machine translation, including phrase-based MT (PBMT) (Felice et al., 2014; Chollampatt, Hoang, and Ng, 2016; Junczys-Dowmunt and Grundkiewicz, 2016) and neural encoder-decoder models (Yuan and Briscoe, 2016). Napoles, Sakaguchi, and Tetreault (2017) recently showed that a neural encoder-decoder can outperform PBMT on a fluency-oriented GEC data and metric.

We investigate training methodologies in the neural encoder-decoder for GEC. To train the neural encoder-decoder models, maximum likelihood estimation (MLE) has been used. As we discuss further details in § 2.2, the objective of MLE is to maximize the (log) likelihood of the parameters for a given training data.

As Ranzato et al. (2015) indicates, however, MLE has drawbacks. The MLE objective is based on *word-level* accuracy against the reference, and the model is not exposed to the predicted output during training (exposure bias). This becomes problematic, because once the model fails to predict a correct word, it falls off the right track and does not come back to it easily.

To address the issues, we employ a neural encoder-decoder GEC model with a rein-

Algorithm 3: Reinforcement learning for neural encoder-decoder model.

Input: Pairs of source (X) and target (Y)
Output: Model parameter $\hat{\theta}$

initialize($\hat{\theta}$)
for $(x, y) \in (X, Y)$ **do**
    $(\hat{y}_1, ...\hat{y}_k), (p(\hat{y}_1), ...p(\hat{y}_k)) = \text{sample}(x, k, \hat{\theta})$
    $p(\hat{y})$ = normalize($p(\hat{y})$)
    $\bar{r}(\hat{y}) = 0$ `// expected reward`
    for $\hat{y}_i \in \hat{y}$ **do**
        $\bar{r}(\hat{y}) += p(\hat{y}_i) \cdot \text{score}(\hat{y}_i, y)$
    backprop($\hat{\theta}, \bar{r}$) `// policy gradient` $\frac{\partial}{\partial \hat{\theta}}$
return $\hat{\theta}$

forcement learning approach in which we directly optimize the model toward our final objective (i.e., evaluation metric). The objective of the neural reinforcement learning model (NRL) is to maximize the expected reward on the training data. In our case, the reward is (sentence-level) GLEU score, which captures fluency as well as error correction accuracy by comparing against reference. The model updates the parameters through back-propagation according to the reward from predicted outputs. The high-level description of the training procedure is shown in Algorithm 3, and more details are elaborated in §7.2. We explain more details in §7.2, mentioning NRL as generalization of minimum risk training (MRT) (Shen et al., 2016). To our knowledge, this is the first attempt to employ reinforcement learning for directly optimizing the encoder-decoder model for GEC task.

We run GEC experiments on a fluency-oriented GEC corpus (§7.3), demonstrating that NRL outperforms the MLE baseline both in human and automated evaluation metrics.

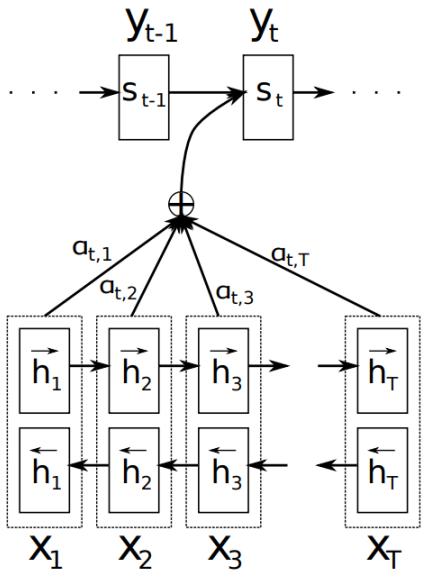

Figure 7.1: Illustrative example of the neural encoder-decoder model with attention mechanism (adapted from Bahdanau, Cho, and Bengio (2014)).

7.2 Model and Optimization

We use the *attentional neural encoder-decoder* model (Bahdanau, Cho, and Bengio, 2014) as a basis for both NRL and MLE. The model takes (possibly ungrammatical) source sentences $x \in X$ as an input, and predicts grammatical and fluent output sentences $y \in Y$ according to the model parameter θ. The model consists of two sub-modules, *encoder* and *decoder*. The encoder transforms x into a sequence of vector representations (hidden states h) using a bidirectional gated recurrent neural network (GRU) (Chung et al., 2014). With the encoder representation, the decoder predicts a word y_t at a time, using previous token y_{t-1} and linear combination c_t of encoder information as attention α.

Thus, the entire encoder-decoder model (with attention mechanism) is formalized as

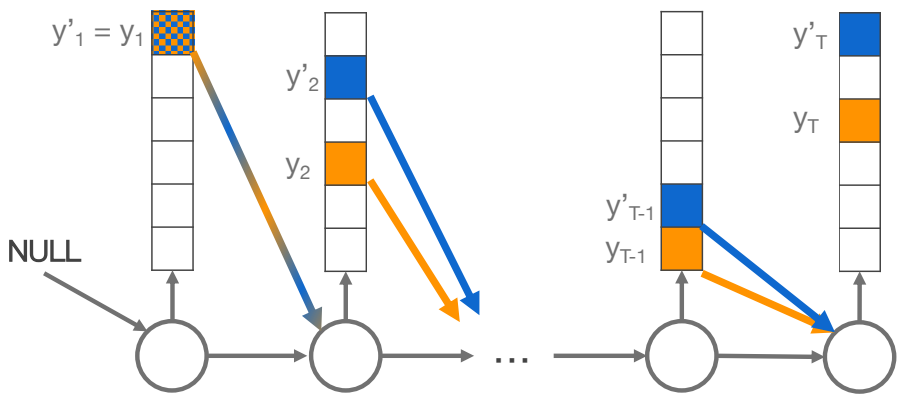

Figure 7.2: Illustration of the exposure bias. The decoder predicts a word conditioned on the sequence (y_1^{t-1}) of the gold labels during training, whereas it does with the predicted (argmax) word sequence ($\hat{y}_1^{t-1}$) at test time.

follows:

$$\log p(y|x;\theta) = \sum_{t=1}^{T} \log p(y_t|x, y_1^{t-1}, c_t;\theta) \tag{7.1}$$

$$c_t = \sum_{s=1}^{S} \alpha_{st} h_s \tag{7.2}$$

$$\alpha_{st} h_s = \frac{\exp(e_{st})}{\sum_S \exp(e_{st})}, \tag{7.3}$$

where e_{st} is called an alignment model which computes how well the hidden states in source position (s) and target position (t) match with each other. (Bahdanau, Cho, and Bengio, 2014) uses a feedforward neural network (i.e., multiple layer perceptron) for the alignment.

7.2.1 Maximum Likelihood Estimation

Maximum Likelihood Estimation training (MLE) is a standard optimization method for encoder-decoder models. In MLE, the objective is to maximize the log likelihood of the correct sequence for a given sequence for the entire training data.

$$L(\theta) = \sum_{\langle X,Y \rangle} \sum_{t=1}^{T} \log p(y_t | x, y_1^{t-1}; \theta) \tag{7.4}$$

The gradient of $L(\theta)$ is as follows:

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_{\langle X,Y \rangle} \sum_{t=1}^{T} \frac{\nabla p(y_t | x, y_1^{t-1}; \theta)}{p(y_t | x, y_1^{t-1}; \theta)} \tag{7.5}$$

One drawback of MLE is the *exposure bias* (Ranzato et al., 2015). The decoder predicts a word conditioned on the correct word sequence (y_1^{t-1}) during training, whereas it does with the predicted (argmax) word sequence ($\hat{y}_1^{t-1}$) at test time. Namely, the model is not exposed to the predicted words in training time. This is problematic, because once the model fails to predict a correct word at test time, it falls off the right track and does not come back to it easily (Figure 7.2). Furthermore, in most sentence generation tasks, the MLE objective does not necessarily correlate with our final evaluation metrics, such as BLEU (Papineni et al., 2002) in machine translation and ROUGE (Lin, 2004) in summarization. This is because MLE optimizes word level predictions at each time step instead of evaluating sentences as a whole.

GEC is no exception. It depends on sentence-level evaluation that considers grammaticality and fluency. For this purpose, it is natural to use GLEU (Napoles et al., 2015), which has been used as a fluency-oriented GEC metric. We explain more details of this metric in §7.2.3.

7.2.2 Neural Reinforcement Learning

To address the issues in MLE, we directly optimize the neural encoder-decoder model toward our final objective for GEC using reinforcement learning. In reinforcement learning, *agents* aim to maximize expected *rewards* by taking *actions* and updating the *policy* under a given *state*. In the neural encoder-decoder model, we treat the encoder-decoder as an agent which predicts a word from a fixed vocabulary at each time step (the action), given the hidden states of the neural encoder-decoder representation. The key difference from MLE is that the reward is not restricted to token-level accuracy. Namely, any arbitrary metric is applicable as the reward.[2]

Since we use GLEU as the final evaluation metric, the objective of NRL is to maximize the expected GLEU by learning the model parameter. Formally, the objective in NRL is defined as follows.

$$\begin{aligned} J(\theta) &= \mathbb{E}[r(\hat{y}, y)] \\ &= \sum_{\hat{y} \in S(x)} p(\hat{y}|x;\theta) r(\hat{y}, y) \end{aligned} \tag{7.6}$$

[2]The reward is given at the end of the decoder output (i.e., delayed reward).

where $S(x)$ is a sampling function that produces k samples $\hat{y}_1, ...\hat{y}_k$, $p(\hat{y}|x;\theta)$ is a probability of the output sentence, and $r(\hat{y}, y)$ is the reward for $\hat{y}_k$ given a reference set y. As described in Algorithm 3, given a pair of source sentence and the reference (x, y), NRL takes k sample outputs $\hat{y}_1$, ... $\hat{y}_k$ and their probabilities $p(\hat{y}_1)$, ... $p(\hat{y}_k)$ (line 3).[3] Then, the expected reward is computed by multiplying the probability and metric score for each sample $\hat{y}_i$ (line 7).

In the encoder-decoder model, the parameters θ are updated through back-propagation and the number of parameter updates is determined by the partial derivative of $J(\theta)$, called the *policy gradient* (Williams, 1992; Sutton et al., 1999) in reinforcement learning:

$$\frac{\partial J(\theta)}{\partial \theta} = \alpha \mathbb{E}\left[\nabla \log p(\hat{y})\{r(\hat{y}, y) - b\}\right] \tag{7.7}$$

where α is a learning rate and b is an arbitrary baseline reward to reduce the variance. The sample mean reward is often used for b (Williams, 1992), and we follow it in NRL.

It is reasonable to compare NRL to minimum risk training (MRT) (Shen et al., 2016). In fact, NRL with a *negative expected reward* can be regarded as MRT. with mini-batch size being 1. The gradient of MRT objective is a special case of *policy gradient* in NRL. We show mathematical details about the relevance between NRL and MRT in §7.2.4.

[3] We sampled sentences from softmax distribution.

7.2.3 Reward in Grammatical Error Correction

To capture fluency as well as grammaticality in evaluation on such references, we use GLEU as the reward. In Chapter 5, we have already shown GLEU to be more strongly preferred than other GEC metrics by native speakers. Similar to BLEU in machine translation, GLEU computes n-gram precision between the system hypothesis (H) and the reference (R). In GLEU, however, n-grams in source (S) are also considered. The precision is penalized when the n-gram in H overlaps with the source and not with the reference. Formally,

$$\text{GLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{4} \frac{1}{n} \log p'_n\right) \tag{7.8}$$

$$p'_n = \frac{N(H,R) - [N(H,S) - N(H,S,R)]}{N(H)} \tag{7.9}$$

$$\text{BP} = \begin{cases} 1 & \text{if } h > r \\ \exp(1 - r/h) & \text{if } h \leq r \end{cases} \tag{7.10}$$

where $N(A, B, C, ...)$ is the number of overlapped n-grams among the sets, and BP *brevity penalty* is compute based on token length in the system hypothesis (h) and the reference (r).

7.2.4 Minimum Risk Training and Policy Gradient in Reinforcement Learning

We explain the relevance between minimum risk training (MRT) (Shen et al., 2016) and neural reinforcement learning (NRL) for training neural encoder-decoder models. We

describe the detailed derivation of gradient in MRT, and show that MRT is a special case of NRL.

As explained briefly in § 7.2, the model takes ungrammatical source sentences $x \in X$ as an input, and predicts grammatical and fluent output sentences $y \in Y$. The objective function in NRL and MRT are written as follows.

$$J(\theta) = \mathbb{E}[r(\hat{y}, y)] \tag{7.11}$$

$$R(\theta) = \sum_{(X,Y)} \mathbb{E}[\Delta(\hat{y}, y)] \tag{7.12}$$

where $r(\hat{y}, y)$ is the *reward* and $\Delta(\hat{y}, y)$ is the *risk* for an output ($\hat{y}$).

For the sake of simplicity, we consider expected loss in MRT for a single training pair:

$$\begin{aligned} \tilde{R}(\theta) &= \mathbb{E}[\Delta(\hat{y}, y)] \\ &= \sum_{\hat{y} \in S(x)} q(\hat{y}|x; \theta, \alpha)\Delta(\hat{y}, y) \end{aligned} \tag{7.13}$$

where

$$q(\hat{y}|x; \theta, \alpha) = \frac{p(\hat{y}|x; \theta)^\alpha}{\sum_{\hat{y}' \in S(x)} p(\hat{y}'|x; \theta)^\alpha} \tag{7.14}$$

$S(x)$ is a sampling function that produces k samples $\hat{y}_1, ...\hat{y}_k$, and α is a smoothing parameter for the samples (Och, 2003). Although the direction to optimize (i.e., minimizing or maximizing) is different, we see the similarity between $J(\theta)$ and $\tilde{R}(\theta)$ in the sense that they both optimize models directly towards evaluation metrics.

The partial derivative of $\tilde{R}(\theta)$ with respect to the model parameter θ is derived as fol-

lows.

$$\begin{aligned}\frac{\partial \tilde{R}(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \sum_{\hat{y} \in S(x)} q(\hat{y}|x;\theta,\alpha)\Delta(\hat{y},y) \\ &= \sum_{\hat{y} \in S(x)} \Delta(\hat{y},y) \frac{\partial}{\partial \theta} q(\hat{y}|x;\theta,\alpha) \end{aligned} \tag{7.15}$$

We need $\frac{\partial}{\partial \theta} q(\hat{y}|x;\theta,\alpha)$ in (7.15). For space efficiency, we use $q(\hat{y})$ as $q(\hat{y}|x;\theta,\alpha)$ and $p(\hat{y})$ as $p(\hat{y}|x;\theta)$ below.

$$\begin{aligned}\frac{\partial}{\partial \theta} q(\hat{y}) &= \frac{\partial q(\hat{y})}{\partial p(\hat{y})} \frac{\partial p(\hat{y})}{\partial \theta} \quad (\because \text{chain rule}) \\ &= \frac{\partial q(\hat{y})}{\partial p(\hat{y})} \nabla p(\hat{y}) \end{aligned} \tag{7.16}$$

For $\frac{\partial q(\hat{y})}{\partial p(\hat{y})}$, by applying the quotient rule to (7.14),

$$\begin{aligned}\frac{\partial q(\hat{y})}{\partial p(\hat{y})} &= \frac{\{\sum_{\hat{y}'} p(\hat{y}')^\alpha\} \frac{\partial}{\partial p(\hat{y})} p(\hat{y})^\alpha - p(\hat{y})^\alpha \frac{\partial}{\partial p(\hat{y})} \sum_{\hat{y}'} p(\hat{y}')^\alpha}{\{\sum_{\hat{y}'} p(\hat{y}')^\alpha\}^2} \\ &= \frac{\alpha p(\hat{y})^{\alpha-1}}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} - \frac{\alpha p(\hat{y})^\alpha p(\hat{y})^{\alpha-1}}{\{\sum_{\hat{y}'} p(\hat{y}')^\alpha\}^2} \\ &= \alpha \frac{p(\hat{y})^{\alpha-1}}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} \left\{ 1 - \frac{p(\hat{y})^\alpha}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} \right\} \\ &= \alpha \frac{p(\hat{y})^\alpha}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} \frac{1}{p(\hat{y})} \left\{ 1 - \frac{p(\hat{y})^\alpha}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} \right\} \end{aligned} \tag{7.17}$$

Corpus	# sents.	mean chars per sent.	# sents. edited
NUCLE	57k	115	38%
FCE	34k	74	62%
Lang-8	1M	56	35%

Table 7.1: GEC corpora for training

Thus, from (7.16) and (7.17), (7.15) is

$$\begin{aligned}\frac{\partial \tilde{R}(\theta)}{\partial \theta} &= \sum_{\hat{y} \in S(x)} \Delta(\hat{y}, y) \nabla p(\hat{y}) \left[\alpha \frac{p(\hat{y})^\alpha}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} \frac{1}{p(\hat{y})} \left\{ 1 - \frac{p(\hat{y})^\alpha}{\sum_{\hat{y}'} p(\hat{y}')^\alpha} \right\} \right] \\ &= \alpha \mathbb{E} \left[\nabla p(\hat{y}) \cdot \frac{1}{p(\hat{y})} \left\{ \Delta(\hat{y}, y) - \mathbb{E} \left[\Delta(\hat{y}, y) \right] \right\} \right] \\ &= \alpha \mathbb{E} \left[\nabla \log p(\hat{y}) \left\{ \Delta(\hat{y}, y) - \mathbb{E} \left[\Delta(\hat{y}, y) \right] \right\} \right] \end{aligned} \tag{7.18}$$

According to the policy gradient theorem for REINFORCE (Williams, 1992; Sutton et al., 1999), the partial derivative of (7.11) is given as follows:

$$\frac{\partial J(\theta)}{\partial \theta} = \tilde{\alpha} \mathbb{E} \left[\nabla \log p(\hat{y}) \{ r(\hat{y}, y) - b \} \right] \tag{7.19}$$

where $\tilde{\alpha}$ is a *learning rate*[4] and b is arbitrary *baseline reward* to reduce the variance of gradients. Finally, we see that the gradient of MRT (7.18) is a special case of policy gradient in REINFORCE (7.19) with $b = \mathbb{E} \left[\Delta(\hat{y}, y) \right]$. It is also interesting to see that the smoothing parameter α works as a part of learning rate ($\tilde{\alpha}$) in NRL.

7.3 Experiments

7.3.1 Data

For training the models (MLE and NRL), we use the following corpora: the NUS Corpus of Learner English (NUCLE) (Dahlmeier, Ng, and Wu, 2013), the Cambridge Learner Corpus First Certificate English (FCE) (Yannakoudakis, Briscoe, and Medlock, 2011), and the Lang-8 Corpus of learner English (Tajiri, Komachi, and Matsumoto, 2012). As described in Chapter 2 (Table 2.3), all these datasets are publicly and freely available, for the purpose of reproducibility. We show the basic information of the corpora again for reference (Table 7.1). In Table 7.1, we exclude some unreasonable edits (comments by editors, incomplete sentences such as URLs, etc.) using regular expressions and setting a maximum token edit distance within 50% of the original length. We also ignore sentences that are longer than 50 tokens or sentences where more than 5% of tokens are out-of-vocabulary (the vocabulary size is 35k). In total, we use 720k pairs of sentences for training (21k from NUCLE, 32k from FCE, and 667k from Lang-8). Spelling errors are corrected in preprocessing with the Enchant open-source spell checking library.[5]

[4] In this appendix, we use $\tilde{\alpha}$ to distinguish it from smoothing parameter α in MRT.

[5] `https://github.com/AbiWord/enchant`

7.3.2 Hyperparameters

For both MLE and NRL, we set the vocabulary size to be 35k for both source and target. Words are represented by a vector with 512 dimensions. Maximum output token length is 50. The size of hidden layer units is 1,000. Gradients are clipped at 1, and beam size during decoding is 5. We regularize the GRU layer with a dropout probability of 0.2.

For MLE we use mini-batches of size 40, and the ADAM optimizer with a learning rate of 10^{-4}. We train the encoder-decoder with MLE for 900k updates, selecting the best model according to the development set evaluation.

For NRL we set the sample size to be 20. We use the SGD optimizer with a learning rate of 10^{-4}. For the *baseline reward*, we use average of sampled reward following Williams (1992). The sentence GLEU score is used as the reward $r(\hat{y}, y)$. Following a similar (but not the same) strategy of the Mixed Incremental Cross-Entropy Reinforce (MIXER) algorithm (Ranzato et al., 2015), we initialize the model by MLE for 600k updates, followed by another 600k updates using NRL, and select the best model according to the development set evaluation. Our NRL is implemented by extending the Nematus toolkit (Sennrich et al., 2017).[6]

7.3.3 Baselines

In addition to our MLE baseline, we compare four leading GEC systems. All the systems are based on SMT, but they take different approaches. The first model, proposed by

[6]NRL code is available at `https://github.com/keisks/nematus/tree/nrl-gleu`

Models	Methods	# sents. (corpora)
CAMB14	Hybrid (rule + PBMT)	155k (NUCLE, FCE, in-house)
AMU	PBMT + GEC-feat.	2.3M (NUCLE, Lang8)
NUS	PBMT + Neural feat.	2.1M (NUCLE, Lang8)
CAMB16	enc-dec (MLE) + unk alignment	1.96M (non-public CLC)
MLE/NRL	enc-dec (MLE/NRL)	720k (NUCLE, Lang8, FCE)

Table 7.2: Summary of baselines, MLE and NRL models.

Felice et al. (2014), uses a combination of a rule-based system and PBMT with language model reranking (referring as CAMB14). Junczys-Dowmunt and Grundkiewicz (2016) proposed a PBMT model that incorporates linguistic and GEC-oriented sparse features (AMU). Another PBMT model, proposed by Chollampatt, Hoang, and Ng (2016), is integrated with neural contextual features (NUS). Finally, Yuan and Briscoe (2016) proposed a neural encoder-decoder model with MLE training (CAMB16). This model is similar to our MLE model, but CAMB16 additionally trains an unsupervised alignment model to handle spelling errors as well as unknown words, and it uses 1.96M sentence pairs extracted from the non-public Cambridge Learner Corpus (CLC). The summary of baselines is shown in Table 7.2.[7]

[7]The four baselines are not tuned toward the same dev set as MLE and NRL. Also, they use different training set (Table 7.2). We compare them just for reference.

	dev set		test set	
Models	Human	GLEU	Human	GLEU
Original	-1.072	38.21	-0.760	40.54
AMU	-0.405	41.74	-0.168	44.85
CAMB14	-0.160	42.81	-0.225	46.04
NUS	-0.131	46.27	-0.249	50.13
CAMB16	-0.117	47.20	-0.164	52.05
MLE	-0.052	48.24	-0.110	52.75
NRL	0.169	**49.82**	0.111	**53.98**
Reference	1.769	55.26	1.565	62.37

Table 7.3: Human evaluation and GLEU evaluation of system outputs on the development and test set. Human evaluation is obtained from TrueSkill, which computes relative skills (performance) among the systems (higher is better).

7.3.4 Evaluation

For evaluation, we use the JFLEG corpus (Heilman et al., 2014; Napoles, Sakaguchi, and Tetreault, 2017), which consists of 1501 sentences (754: dev, 747: test) with four fluency-oriented references.

In addition to the automated metric (GLEU), we run a human evaluation using Amazon Mechanical Turk (MTurk). We randomly select 200 sentences each from the dev and test set. For each sentence, two turkers are repeatedly asked to rank five systems randomly selected from all eight: the four baseline models, MLE, NRL, one randomly selected human correction, and the original sentence. We infer the evaluation scores by comparing pairwise rankings with the TrueSkill algorithm (Herbrich, Minka, and Graepel, 2006; Sakaguchi, Post, and Van Durme, 2014).

Models	Precision	Recall	M^2 ($F_{0.5}$)
AMU	**69.95**	18.81	45.32
CAMB14	65.09	22.84	47.51
NUS	69.59	29.19	54.50
CAMB16	64.35	32.26	53.67
MLE	66.00	34.62	55.87
NRL	65.93	**37.28**	**57.15**

Table 7.4: M^2 ($F_{0.5}$) scores on the dev set.

Models	Precision	Recall	M^2 ($F_{0.5}$)
AMU	**69.39**	20.79	47.29
CAMB14	63.52	23.44	47.33
NUS	68.08	32.30	55.73
CAMB16	65.66	35.93	56.34
MLE	65.19	37.66	56.88
NRL	65.80	**40.96**	**58.68**

Table 7.5: M^2 ($F_{0.5}$) scores on the test set.

7.3.5 Results

Table 7.3 shows the human evaluation by TrueSkill and automated metric (GLEU). In both dev and test set, NRL outperforms MLE and other baselines in both the human and automatic evaluations. Human evaluation and GLEU scores correlate highly, corroborating the reliability of GLEU. With respect to inter-annotator agreement, Spearman's rank correlation between Turkers is 55.6 for the dev set and 49.2 for the test set. The correlations are sufficiently high to show the agreement between Turkers, considering the low chance level (i.e., ranking five randomly selected systems consistently between two Turkers).

Table 7.4 and 7.5 show the M^2 ($F_{0.5}$) scores (Dahlmeier and Ng, 2012b), which compute phrase-level edits between the system hypothesis and source and compare them with the oracle edits. Although this metric has several drawbacks such as underestimation of system

	NRL> MLE	NRL= MLE	NRL< MLE
Dev	33%	45%	22%
Test	30%	57%	13%

Table 7.6: Ratio of pairwise (preference) judgments between NRL and MLE. NRL>MLE: NRL correction is preferred over MLE. NRL<MLE: MLE is preferred over NRL. NRL=MLE: NRL and MLE are tied.

Orig.	but found that successful people use the people money and use there idea for a way to success .
Ref.	But it was found that successful people use other people 's money and use their ideas as a way to success .
MLE	But found that successful people use the people money and use it for a way to success .
NRL	But found that successful people use the people 's money and use their idea for a way to success .
Orig.	Fish firming uses the lots of special products such as fish meal .
Ref.	Fish firming uses a lot of special products such as fish meal .
MLE	Fish contains a lot of special products such as fish meals .
NRL	Fish shops use the lots of special products such as fish meal .

Table 7.7: Example outputs by MLE and NRL

performance and indiscrimination between "no change" and "wrong edits" (Felice et al., 2014), we see that the correlation between the M^2 scores and human evaluation is still high in the result.

Finally, Table 7.6 shows the percentages of preference in the pairwise comparisons between NRL and MLE. In both the dev and test sets, around 30% of NRL corrections are preferred over MLE and approximately 50% are tied.

7.3.6 Analysis

Table 7.7 presents example outputs from MLE and NRL. In the first example, both MLE and NRL successfully corrected the homophone error (*there vs. their*), but MLE changed the meaning of the original sentence by replacing *their idea* to *it*. Meanwhile, NRL made the sentence more grammatical by adding a possessive *'s*. The second example demonstrates challenging issues for future work in GEC. The correction by MLE looks fairly fluent as well as grammatical, but it is semantically nonsense. The correction by NRL is also fairly fluent and makes sense, but the meaning has been changed too much. For further improvement, better GEC models that are aware of the context or possess world knowledge are needed.

7.4 Summary

We have presented a neural encoder-decoder model with reinforcement learning for GEC. To alleviate the MLE issues (exposure bias and token-level optimization), NRL learns the policy (model parameters) by directly optimizing toward the final objective by treating the final objective as the reward for the encoder-decoder agent. Using a GEC-specific metric, GLEU, we have demonstrated that NRL outperforms the MLE baseline on the fluency-oriented GEC corpus both in human and automated evaluation metrics. As a supplement, we have explained the relevance between minimum risk training (MRT) and NRL, claiming that MRT is a special case of NRL.

Chapter 8

Conclusions and Future Directions

In this thesis, I have discussed automated grammatical error correction (GEC) in terms of the scope (i.e., error types), evaluation metrics, benchmarking datasets, and methods. Specifically, I have:

- presented the difference between *grammaticality* and *fluency*. Both are important for people to communicate smoothly in natural languages. While *grammaticality* is derived from a set of syntactic rules, *fluency* is a level of mastery that goes beyond knowledge of how to follow the rules.

- proposed a robust word recognition model for character-level errors. Motivated by the *Cmabrigde Uinervtisy* (*Cambridge University*) effect (or typoglycemia) in psycholinguistics literature, the model is built on a recurrent neural network. The model shows the cognitive plausibility and significantly more robust performance in word spelling correction compared to existing spell-checkers and character-based convo-

lutional neural networks.

- presented a parsing algorithm that can correct token-level grammatical errors and parse dependency structure jointly. The algorithm shows dependency accuracy and grammaticality improvements for ungrammatical sentences.

- shown the limitations in the conventional GEC annotation and evaluation scheme, which depends on fine-grained error codes, and proposed a novel framework that considers *fluency* as well as *grammaticality*. Compared with conventional error-coded minimal edits, whole-sentence fluency edits have shown higher correlation to human judgments.

- built two datasets for benchmarking fluency-oriented GEC. First, we built fluency edits for the NUCLE corpus. Second, we created the **J**HU **FL**uency-**E**xtended **G**UG corpus (JFLEG) in order to avoid over-reliance on a single benchmark dataset.

- proposed a neural encoder-decoder model with reinforcement learning for sentence-level GEC. The NRL model directly optimizes toward a task-specific evaluation metric, avoiding the exposure bias issue. We ran GEC experiments on a fluency-oriented GEC corpus, demonstrating that NRL outperforms the MLE baseline both in human and automated evaluation metrics.

Although I steadily made strides forward, there are several challenging questions remaining in GEC. Before I conclude the thesis, I will discuss the challenges that need to be

Corpus	Num. Sentences	Num. References	Sentence Changed	Error Type labeld	Fluency edits	Diverse proficiency	Diverse topic	Diverse L1
NUCLE	59k	2	38%	✓	(✗)	✗	✗	✗
FCE	34k	1	62%	✓	✗	✓	✓	✓
Lang-8	2.5M	≥1	42%	✗	✓	✓	✓	✓
JFLEG	1.5k	4	86%	✗	✓	✓	✓	✓

Table 8.1: GEC corpora available for free (for research purposes) and desired properties. ✓ and ✗ indicate whether the corpus exhibits each property. This is the extended version of Table 2.3 in Chapter 2 to which the JFLEG row was added.

addressed and provide recommendations for the field to continue to make further progress. I look at the challenges from three different perspectives: datasets, metrics, and models.[1]

First, as we saw in Chapter 5, the majority of the commonly used datasets are limited to students, specifically college-level ESL writers. To date, the overwhelming majority of publications benchmark on NUCLE, save for a few exceptions such as Cahill et al. (2013) and Rei and Yannakoudakis (2016), which means that research efforts are becoming over-optimized for one set. This lack of diversity means that it is not clear how systems perform on other genres under different training conditions. In this thesis, we built a new GEC corpus for evaluation, the **J**HU **FL**uency-**E**xtended **G**UG corpus (JFLEG) to address this issue. However, we will still suffer from the overfitting problem unless we keep building additional GEC corpora for both training and evaluation. We should look to the parsing community as a warning sign. For well over a decade, the field was heavily focused on improving parsing accuracy on the Penn Treebank (Marcus, Marcinkiewicz, and Santorini,

[1] Some of the following contents were originally published by Sakaguchi, Napoles, and Tetreault (2017).

1993), but robustness was greatly improved with the advent of Ontonotes (Hovy et al., 2006) and the Google Web Treebank (Petrov and McDonald, 2012).

Another issue in the GEC dataset is the size for training, as we saw in Chapter 2. The sister field of machine translation (MT) usually has datasets in the orders of millions or even tens of millions of sentence pairs. The largest GEC datasets barely approach that figure, with 2.5 million sentences at maximum, a number that includes sentences that were not corrected. In Table 8.1, I summarize the strengths and weaknesses of the most commonly used GEC corpora across different properties ranging from size to diversity in native language (L1). The most notable weakness across corpora is the lack of multiple reference corrections. NUCLE contains two corrections per sentence, and JFLEG 4. M^2 and GLEU scores increase with more references but at a diminishing rate (Bryant and Ng, 2015; Sakaguchi et al., 2016). Further investigation is warranted to determine the an ideal number of references, given the trade-off between cost and reliability. NUCLE contains little diversity in proficiency, topic, and learners' L1.

Looking into the future, as the world's communication is not limited to college-level essays, it is important that we have datasets that better represent as much breadth as possible. Ideally, datasets should span various genres (such as e-mails, blog posts, and official documents) and include content from both native and nonnative speakers, as well as from different proficiency levels. All of these changes will enable the field to better assess how we are helping *more* of the world's writers under different conditions, as well as enable one to test adaptation between domains.

Second, as I stated in Chapter 5, I envision evaluation metrics that check that corrections are not only grammatically valid but also native-sounding (i.e., *fluent* and preserve the original meaning or intent of the writer). Although I have shown that two evaluation metrics, M^2 and GLEU, have a strong positive correlation with human judgments, future metrics should be easier to compute and more interpretable. One challenging problem is that all the current metrics highly depend on the references annotated by native speakers. In other words, these metrics assume that the reference set covers all the possible corrections. Of course, it is almost impossible to collect (i.e., annotate) all the possible corrections, even though I use non-error-coded *fluency edits* with crowdsourcing, as shown in Chapter 5. To address this issue, one potential research direction is to look for "reference-less" evaluation metrics. In fact, some studies have started investigating and shown the potential of this approach (Napoles, Sakaguchi, and Tetreault, 2016; Asano, Mizumoto, and Inui, 2017; Choshen and Abend, 2018b), although a lot more needs to be explored, such as sensitivity to different error types.

Another challenge is document level evaluation that considers the global context of the document. Most evaluation schemes to date have focused on the sentence as the minimal unit. It would be good to take the entire document into account and allow for more global rewrites, such as consistent tense.

Ultimately, a metric should say whether or not a system has attained the same level of performance as a human judge. One way of doing this is through a GEC Turing Test, where system outputs are blindly judged alongside human corrections of the same sen-

tences. If human adjudicators think the system outputs are indistinguishable in quality from the human corrections (for example, given a set of criteria such as being good corrections, preserving meaning, and native-sounding), then that is a very strong signal that GEC has attained human-level performance.

Third, regarding the GEC models, the community should have some consensus about data splits (i.e., training, development, and test set). Although I have developed a new fundamental framework for metrics and dataset for evaluation (Chapter 5), the models are trained on different datasets. In other words, we cannot tell whether the improvement actually comes from the model (i.e., algorithm), different training data (as well as different size), or different preprocessing procedures. This is partly due to the lack of a large-scale GEC corpus.[2] In order to compare the models objectively, the community should share (at least one) common data splits, which are publicly and freely available.

Finally, it is important to remember that initial approaches to GEC seemed to focus on providing feedback to English language learners, where specific error types would be targeted and feedback would be given in terms of detection or possible corrections. The work was also motivated by concurrent work in using NLP for automatic essay scoring (for example, Attali and Burstein (2006)).

Chodorow et al. (2012) noted several other applications for GEC: improving overall writing quality for both native and nonnative writers, assistive language learning, and applications within NLP (such as post-editing in MT). More recently, the field has drifted

[2]Recent work by Junczys-Dowmunt et al. (2018) points out that GEC is a low-resource machine translation task.

to "whole-sentence GEC" using statistical or neural MT approaches. In this situation, the writer simply gets a complete rewrite of his or her sentence, which may be useful as an instructional tool in some circumstances, but not all.

There is no consensus on what the focus application(s) should be. The application determines the methods and the evaluation metrics one uses. For example, if one wants to provide feedback to language learners, then a high-precision, interpretable method is preferred.

Conversely, if the application is simply to automatically clean up one's writing without any feedback, then a whole-sentence approach may be preferred. Very few papers delve into error detection and correction for goals other than whole-sentence error correction or targeted feedback for ESL writers. As shown in this thesis, datasets and metrics should be created and determined with a specific goal in mind (e.g., "whole-sentence" error correction for *fluency* as well as *grammaticality*). Thus, the field should keep reassessing the goals and how we evaluate with respect to these goals.

Appendix A

Efficient Elicitation of Annotations for Manual System[1]

As explained in Chapter 5, human judgments are the ideal (reliable) metric to rank natural language generation (NLG) systems, such as machine translation, grammatical error correction, and summarization. In addition, human judgments are useful to objectively compare automated evaluation metrics by their correlation. For example, the better automated metric should correlate with human judgment more closely than the other metrics. Our concern here is the cost (and time) for collecting human judgments.

In this appendix, we discuss how we can efficiently collect human judgments to rank NLG systems. We specifically focus on ranking machine translation systems as a use case.. Of course, the method is applicable to other natural language generation tasks, such as

[1]Much of this appendix was originally published in Sakaguchi, Post, and Van Durme (2014).

GEC systems (Napoles et al., 2015; Grundkiewicz, Junczys-Dowmunt, and Gillian, 2015), dialog systems (Novikova, Dušek, and Rieser, 2017), and summarization (Wolfe et al., 2018).

A.1 Introduction

The Workshop on Statistical Machine Translation (WMT) has long been a central event in the machine translation (MT) community for the evaluation of MT output. It hosts an annual set of shared translation tasks focused mostly on the translation of western European languages. One of its main functions is to publish a ranking of the systems for each task, which are produced by aggregating a large number of human judgments of sentence-level pairwise rankings of system outputs. While the performance on many automatic metrics is also reported (e.g., BLEU (Papineni et al., 2002)), the human evaluation is considered primary, and is in fact used as the gold standard for its metrics task, where evaluation metrics are evaluated.

In machine translation, the longstanding disagreements about evaluation measures do not go away when moving from automatic metrics to human judges. This is due in no small part to the inherent ambiguity and subjectivity of the task, but also arises from the particular way that the WMT organizers produce the rankings. The system-level rankings are produced by collecting pairwise sentence-level comparisons between system outputs. These are then aggregated to produce a complete ordering of all systems, or, more recently, a

#	score	range	system
1	0.638	1	UEDIN-HEAFIELD
2	0.604	2-3	UEDIN
	0.591	2-3	ONLINE-B
4	0.571	4-5	LIMSI-SOUL
	0.562	4-5	KIT
	0.541	5-6	ONLINE-A
7	0.512	7	MES-SIMPLIFIED
8	0.486	8	DCU
9	0.439	9-10	RWTH
	0.429	9-11	CMU-T2T
	0.420	10-11	CU-ZEMAN
12	0.389	12	JHU
13	0.322	13	SHEF-WPROA

Table A.1: System rankings presented as clusters (WMT13 French-English competition). The *score* column is the percentage of time each system was judged better across its comparisons (§A.2.1).

partial ordering (Koehn, 2012), with systems clustered where they cannot be distinguished in a statistically significant way (Table A.1, taken from Bojar et al. (2013)).

A number of problems have been noted with this approach. The first has to do with the nature of ranking itself. Over the past few years, the WMT organizers have introduced a number of minor tweaks to the ranking algorithm (§A.2) in reaction to largely intuitive arguments that have been raised about how the evaluation is conducted (Bojar et al., 2011; Lopez, 2012). While these tweaks have been sensible (and later corroborated), Hopkins and May (2013) point out that this is essentially a model selection task, and should properly be driven by empirical performance on held-out data according to some metric. Instead of intuition, they suggest perplexity, and show that a novel graphical model outperforms existing approaches on that metric, with less amount of data.

A second problem is the deficiency of the models used to produce the ranking, which

work by computing simple ratios of wins (and, optionally, ties) to losses. Such approaches do not consider the relative difficulty of system matchups, and thus leave open the possibility that a system is ranked highly from the luck of comparisons against poorer opponents.

Third, a large number of judgments need to be collected in order to separate the systems into clusters to produce a partial ranking. The sheer size of the space of possible comparisons (all pairs of systems times the number of segments in the test set) requires sampling from this space and distributing the annotations across a number of judges. Even still, the number of judgments needed to produce statistically significant rankings like those in Table A.1 grows quadratically in the number of participating systems (Koehn, 2012), often forcing the use of paid, lower-quality annotators hired on Amazon's Mechanical Turk. Part of the problem is that the sampling strategy collects data uniformly across system pairings. Intuitively, we should need many fewer annotations between systems with divergent base performance levels, instead focusing the collection effort on system pairs whose performance is more matched, in order to tease out the gaps between similarly-performing systems. Why spend precious human time on redundantly affirming predictable outcomes?

To address these issues, we developed a variation of the TrueSkill model (Herbrich, Minka, and Graepel, 2006), an adaptive model of competitions originally developed for the Xbox Live online gaming community. It assumes that each player's skill level follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, in which μ represents a player's mean performance, and σ^2 the system's uncertainty about its current estimate of this mean. These values are updated after each "game" (in our case, the value of a ternary judgment) in proportion to how

surprising the outcome is. TrueSkill has been adapted to a number of areas, including chess, advertising, and academic conference management.

The rest of this appendix provides an empirical comparison of a number of models of human evaluation. We evaluate on perplexity and also on accuracy, showing that the two are not always correlated, and arguing for the primacy of the latter. We find that TrueSkill outperforms other models. Moreover, TrueSkill also allows us to drastically reduce the amount of data that needs to be collected by sampling non-uniformly from the space of all competitions, which also allows for greater separation of the systems into ranked clusters.

A.2 Models

Before introducing our adaptation of the TrueSkill model for ranking translation systems with human judgments (§A.2.3), we describe two comparisons: the "Expected Wins" model used in recent evaluations, and the Bayesian model proposed by Hopkins and May (§A.2.2).

As we described briefly in the introduction, WMT produces system rankings by aggregating sentence-level ternary judgments of the form:

$$(i, S_1, S_2, \pi)$$

where i is the source segment (id), S_1 and S_2 are the system pair drawn from a set of systems $\{S\}$, and $\pi \in \{<, >, =\}$ denotes whether the first system was judged to be better

than, worse than, or equivalent to the second. These ternary judgments are obtained by presenting judges with a randomly-selected input sentence and the reference, followed by *five* randomly-selected translations of that sentence. Annotators are asked to rank these systems from best (rank 1) to worst (rank 5), ties permitted, and with no meaning ascribed to the absolute values or differences between ranks. This is done to accelerate data collection, since it yields ten pairwise comparisons per ranking. Tens of thousands of judgments of this form constitute the raw data used to compute the system-level rankings. All the work described in this section is computed over these pairwise comparisons, which are treated as if they were collected independently.

A.2.1 Expected Wins

The "Expected Wins" model computes the percentage of times that each system wins in its pairwise comparisons. Let A be the complete set of annotations or judgments of the form $\{i, S_1, S_2, \pi_R\}$. We assume these judgments have been converted into a normal form where S_1 is either the winner or is tied with S_2, and therefore $\pi_R \in \{<, =\}$. Let $\delta(x, y)$ be the Kronecker delta function.[2] We then define the function:

$$\text{wins}(S_i, S_j) = \sum_{n=1}^{|A|} \delta(S_i, S_1^{(n)})\delta(S_j, S_2^{(n)})\delta(\pi_R^{(n)}, <) \tag{A.1}$$

[2] $\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{o.w.} \end{cases}$

which counts the number of annotations for which system S_i was ranked better than system S_j. We define a single-variable version that marginalizes over all annotations:

$$\text{wins}(S_i) = \sum_{S_j \neq S_i} \text{wins}(S_i, S_j) \tag{A.2}$$

We also define analogous functions for *loses* and *ties*. Until the WMT11 evaluation (Callison-Burch et al., 2011), the score for each system S_i was computed as follows:

$$\text{score}(S_i) = \frac{\text{wins}(S_i) + \text{ties}(S_i)}{\text{wins}(S_i) + \text{ties}(S_i) + \text{loses}(S_i)} \tag{A.3}$$

Bojar et al. (2011) suggested that the inclusion of ties biased the results, due to their large numbers, the underlying similarity of many of the models, and the fact that they are counted for both systems in the tie, and proposed the following modified scoring function:

$$\text{score}(S_i) = \frac{1}{|\{S\}|} \sum_{S_j \neq S_i} \frac{\text{wins}(S_i, S_j)}{\text{wins}(S_i, S_j) + \text{wins}(S_j, S_i)} \tag{A.4}$$

This metric computes an average relative frequency of wins, excluding ties, and was used in WMT12 and WMT13 (Callison-Burch et al., 2012; Bojar et al., 2013).

The decision to exclude ties isn't without its problems; for example, an evaluation where two systems are nearly always judged equivalent should be relevant in producing the final ranking of systems. Furthermore, as Hopkins and May (2013) point out, throwing out data to avoid biasing a model suggests a problem with the model. We now turn to a

description of their model, which addresses these problems.

A.2.2 The Hopkins and May (2013) model

Recent papers (Koehn, 2012; Hopkins and May, 2013) have proposed models focused on the *relative ability* of the competition systems. These approaches assume that each system has a mean quality represented by a Gaussian distribution with a fixed variance shared across all systems. In the graphical model formulation of Hopkins and May (2013), the pairwise judgments (i, S_1, S_2, π) are imagined to have been generated according to the following process:

- Select a source sentence i
- Select two systems S_1 and S_2. A system S_j is associated with a Gaussian distribution $\mathcal{N}(\mu_{S_j}, \sigma_a^2)$, samples from which represent the quality of translations
- Draw two "translations", adding random Gaussian noise with variance σ_{obs}^2 to simulate the subjectivity of the task and the differences among annotators:

$$q_1 \sim \mathcal{N}(\mu_{S_1}, \sigma_a^2) + \mathcal{N}(0, \sigma_{obs}^2) \tag{A.5}$$

$$q_2 \sim \mathcal{N}(\mu_{S_2}, \sigma_a^2) + \mathcal{N}(0, \sigma_{obs}^2) \tag{A.6}$$

- Let d be a nonzero real number that defines a fixed decision radius. Produce a rating

π according to:[3]

$$\pi = \begin{cases} < & q_1 - q_2 > d \quad \text{(A.7a)} \\ > & q_2 - q_1 > d \quad \text{(A.7b)} \\ = & \text{otherwise} \quad \text{(A.7c)} \end{cases}$$

The task is to then infer the posterior parameters, given the data: the system means μ_{S_j} and, by necessity, the latent values $\{q_i\}$ for each of the pairwise comparison training instances. Hopkins and May do not publish code or describe details of this algorithm beyond mentioning Gibbs sampling, so we used our own implementation,[4] and describe it here for completeness.

After initialization, we have training instances of the form $(i, S_1, S_2, \pi_R, q_1, q_2)$, where all but the q_i are observed. At a high level, the sampler iterates over the training data, inferring values of q_1 and q_2 for each annotation together in a single step of the sampler from the current values of the systems means, $\{\mu_j\}$.[5] At the end of each iteration, these means are then recomputed by re-averaging all values of $\{q_i\}$ associated with that system. After the burn-in period, the μs are stored as samples, which are averaged when the sampling concludes.

[3]Note that better systems have *higher* relative abilities $\{\mu_{S_j}\}$. Better translations subsequently have on-average higher values $\{q_i\}$, which translate into a *lower* ranking π.

[4]`github.com/keisks/wmt-trueskill`

[5]This worked better than a version of the sampler that changed one at a time.

During each iteration, q_1 and q_2 are resampled from their corresponding system means:

$$\begin{aligned} q_1 &\sim \mathcal{N}(\mu_{S_1}, \sigma_a^2) \\ q_2 &\sim \mathcal{N}(\mu_{S_2}, \sigma_a^2) \end{aligned} \tag{A.8}$$

We then update these values to respect the annotation π as follows. Let $t = q_1 - q_2$ (S_1 is the winner by human judgments), and ensure that the values are outside the decision radius, d:

$$q_1' = \begin{cases} q_1 & t \geq d \quad \text{(A.9a)} \\ q_1 + \frac{1}{2}(d - t) & \text{otherwise} \quad \text{(A.9b)} \end{cases}$$

$$q_2' = \begin{cases} q_2 & t \geq d \quad \text{(A.10a)} \\ q_2 - \frac{1}{2}(d - t) & \text{otherwise} \quad \text{(A.10b)} \end{cases}$$

In the case of a tie:

$$q_1' = \begin{cases} q_1 + \frac{1}{2}(d - t) & t \geq d \quad \text{(A.11a)} \\ q_1 & t < d \quad \text{(A.11b)} \\ q_1 + \frac{1}{2}(-d - t) & t \leq -d \quad \text{(A.11c)} \end{cases}$$

$$q_2' = \begin{cases} q_2 - \frac{1}{2}(d - t) & t \geq d \quad \text{(A.12a)} \\ q_2 & t < d \quad \text{(A.12b)} \\ q_2 - \frac{1}{2}(-d - t) & t \leq -d \quad \text{(A.12c)} \end{cases}$$

These values are stored for the current iteration and averaged at its end to produce new estimates of the system means. The quantity $d - t$ can be interpreted as a *loss function*, returning a high value when the observed outcome is unexpected and a low value otherwise (Figure A.1).

A.2.3 TrueSkill

Prior to 2012, the WMT organizers included reference translations among the system comparisons. These were used as a control against which the evaluators could be measured for consistency, on the assumption that the reference was almost always best. They were also included as data points in computing the system ranking. Another of Bojar et al. (2011)'s suggestions was to exclude this data, because systems compared more often against the references suffered unfairly. This can be further generalized to the observation that not all competitions are equal, and a good model should incorporate some notion of "match difficulty" when evaluating system's abilities. The inference procedure above incorporates this notion implicitly in the inference procedure, but the model itself does not include a notion of match difficulty or outcome surprisal.

A model that does is TrueSkill[6] (Herbrich, Minka, and Graepel, 2006). TrueSkill is an adaptive, online system that also assumes that each system's skill level follows a Gaussian distribution, maintaining a mean μ_{S_j} for each system S_j representing its current estimate of

[6]The goal of this section is to provide an intuitive description of TrueSkill as adapted for WMT manual evaluations, with enough detail to carry the main ideas. For more details, please see Herbrich, Minka, and Graepel (2006).

that system's native ability. However, it also maintains a per-system variance, $\sigma^2_{S_j}$, which represents TrueSkill's uncertainty about its estimate of each mean. After an outcome is observed (a game in which the result is a win, loss, or draw), the size of the updates is proportional to how surprising the outcome was, which is computed from the current system means and variances. If a translation from a system with a high mean is judged better than a system with a greatly lower mean, the result is not surprising, and the update size for the corresponding system means will be small. On the other hand, when an upset occurs in a competition, the means will receive larger updates.

Before defining the update equations, we need to be more concrete about how this notion of surprisal is incorporated. Let $t = \mu_{S_1} - \mu_{S_2}$, the difference in system relative abilities, and let ϵ be a fixed hyper-parameter corresponding to the earlier decision radius. We then define two loss functions of this difference for wins and for ties:

$$\begin{aligned} v_{\text{win}}(t,\epsilon) &= \frac{\mathcal{N}(-\epsilon + t)}{\Phi(-\epsilon + t)} \\ v_{\text{tie}}(t,\epsilon) &= \frac{\mathcal{N}(-\epsilon - t) - \mathcal{N}(\epsilon - t)}{\Phi(\epsilon - t) - \Phi(-\epsilon - t)} \end{aligned} \tag{A.13}$$

where $\Phi(x)$ is the cumulative distribution function and the $\mathcal{N}$s are Gaussians. Figures A.1 and A.2 display plots of these two functions compared to the Hopkins and May model. Note how v_{win} (Figure A.1) increases exponentially as μ_{S_2} becomes greater than the (purportedly) better system, μ_{S_1}.

As noted above, TrueSkill maintains not only estimates $\{\mu_{S_j}\}$ of system abilities, but

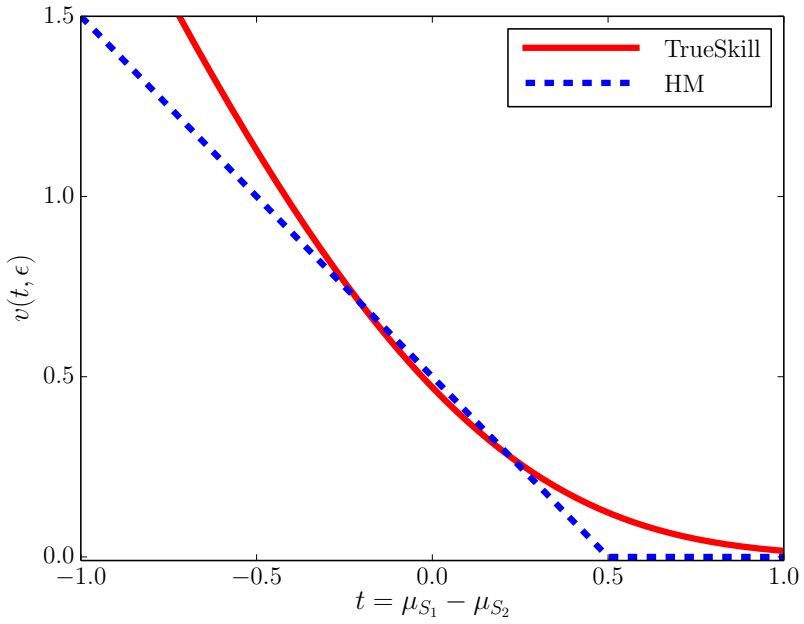

Figure A.1: TrueSkill's v_{win} and the corresponding *loss function* in the Hopkins and May model as a function of the difference t of system means ($\epsilon = 0.5, c = 0.8$ for TrueSkill, and $d = 0.5$ for Hopkins and May model).

also system-specific confidences about those estimates $\{\sigma_{S_j}\}$. These confidences also factor into the updates: while surprising outcomes result in larger updates to system means, higher confidences (represented by smaller variances) result in *smaller* updates. TrueSkill defines the following value:

$$c^2 = 2\beta^2 + \sigma_{S_1}^2 + \sigma_{S_2}^2 \tag{A.14}$$

which accumulates the variances along β, another free parameter. We can now define the

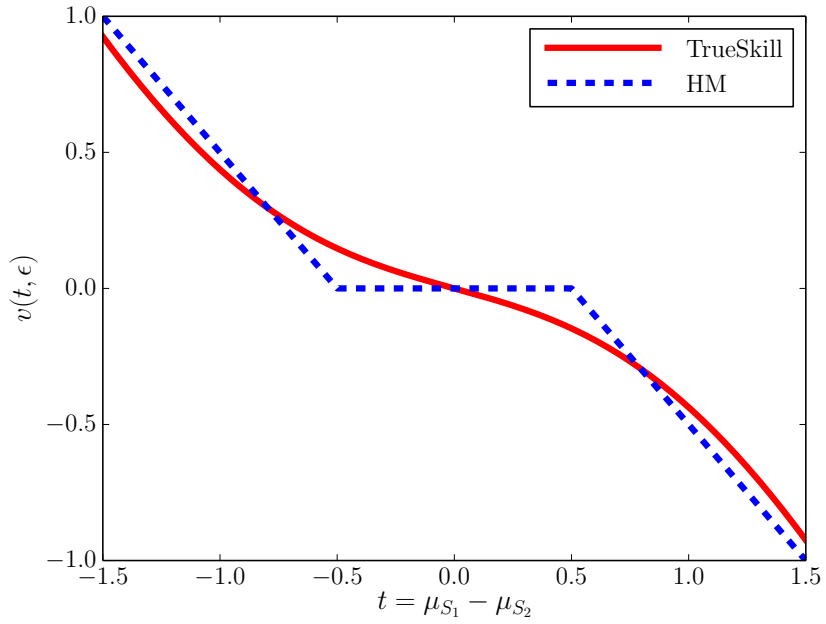

Figure A.2: TrueSkills v_{tie} and the corresponding *loss function* in the Hopkins and May model as a function of the difference t of system means ($\epsilon = 0.5, c = 0.3$, and $d = 0.5$).

update equations for the system means:

$$\mu_{S_1} = \mu_{S_1} + \frac{\sigma_{S_1}^2}{c} \cdot v\left(\frac{t}{c}, \frac{\epsilon}{c}\right) \tag{A.15}$$

$$\mu_{S_2} = \mu_{S_2} - \frac{\sigma_{S_2}^2}{c} \cdot v\left(\frac{t}{c}, \frac{\epsilon}{c}\right) \tag{A.16}$$

The second term in these equations captures the idea about balancing surprisal with confidence, described above.

In order to update the system-level confidences, TrueSkill defines another set of functions, w, for the cases of wins and ties. These functions are multiplicative factors that affect the amount of change in σ^2:

$$w_{\text{win}}(t,\epsilon) = v_{\text{win}} \cdot (v_{\text{win}} + t - \epsilon) \tag{A.17}$$

$$w_{\text{tie}}(t,\epsilon) = v_{\text{tie}} + \frac{(\epsilon - t) \cdot \mathcal{N}(\epsilon - t) + (\epsilon + t) \cdot \mathcal{N}(\epsilon + t)}{\Phi(\epsilon - t) - \Phi(-\epsilon - t)} \tag{A.18}$$

The underlying idea is that these functions capture the outcome surprisal via v. This update always decreases the size of the variances σ^2, which means uncertainty of μ decreases as comparisons go on. With these defined, we can conclude by defining the updates for $\sigma^2_{S_1}$ and $\sigma^2_{S_2}$:

$$\sigma^2_{S_1} = \sigma^2_{S_1} \cdot \left[1 - \frac{\sigma^2_{S_1}}{c^2} \cdot w\left(\frac{t}{c}, \frac{\epsilon}{c}\right)\right] \tag{A.19}$$

$$\sigma^2_{S_2} = \sigma^2_{S_2} \cdot \left[1 - \frac{\sigma^2_{S_2}}{c^2} \cdot w\left(\frac{t}{c}, \frac{\epsilon}{c}\right)\right] \tag{A.20}$$

One final complication not presented here but relevant to adapting TrueSkill to the WMT setting: the parameter β and another parameter (not discussed) τ are incorporated into the update equations to give more weight to recent matches. This "latest-oriented" property is useful in the gaming setting for which TrueSkill was built, where players improve over time, but is not applicable in the WMT competition setting. To cancel this property in TrueSkill, we set $\tau = 0$ and $\beta = 0.025 \cdot |A| \cdot \sigma^2$ in order to lessen the impact of the order in which annotations are presented to the system.

A.2.4 Data selection with TrueSkill

A drawback of the standard WMT data collection method is that it samples uniformly from the space of pairwise system combinations. This is undesirable: systems with vastly divergent relative ability need not be compared as often as systems that are more evenly matched. Unfortunately, one cannot sample non-uniformly without knowing ahead of time which systems are better. TrueSkill provides a solution to this dilemma with its *match-selection* ability: systems with similar means and low variances can be confidently considered to be close matches. This presents a strong possibility of reducing the amount of data that needs to be collected in the WMT competitions. In fact, the TrueSkill formulation provides a way to compute the probability of a draw between two systems, which can be used to compute for a system S_i a conditional distribution over matches with other systems $\{S_{j\neq i}\}$.

Formally, in the TrueSkill model, the *match-selection* (chance to draw) between two players (systems in WMT) is computed as follows:

$$p_{\text{draw}} = \sqrt{\frac{2\beta^2}{c^2}} \cdot \exp(-\frac{(\mu_a - \mu_b)^2}{2c^2}) \tag{A.21}$$

However, our setting for canceling the "latest-oriented" property affects this matching quality equation, where most systems are almost equally competitive (≈ 1). Therefore, we

modify the equation in the following manner which simply depends on the difference of μ.

$$\hat{p}_{\text{draw}} = \frac{1}{\exp(|\mu_a - \mu_b|)} \tag{A.22}$$

TrueSkill selects the matches it would like to create, according to this selection criteria. We do this according to the following process:

1. Select a system S_1 (e.g., the one with the highest variance)
2. Compute a normalized distribution over matches with other systems pairs $\hat{p}_{\text{draw}}$
3. Draw a system S_2 from this distribution
4. Draw a source sentence, and present to the judge for annotation

A.3 Experimental setup

A.3.1 Datasets

We used the evaluation data released by WMT13.[7] The data contains (1) five-way system rankings made by either researchers or Turkers and (2) translation data consisting of source sentences, human reference translations, and submitted translations. Data exists for 10 language pairs. More details about the dataset can be found in the WMT 2013 overview paper (Bojar et al., 2013).

[7] statmt.org/wmt13/results.html

Each five-way system ranking was converted into ten pairwise judgments (§A.2). We trained the models using randomly selected sets of 400, 800, 1,600, 3,200, and 6,400 pairwise comparisons, each produced in two ways: selecting from all researchers, or split between researchers and Turkers. An important note is that the training data differs according to the model. For the Expected Wins and Hopkins and May model, we simply sample uniformly at random. The TrueSkill model, however, selects its own training data (with replacement) according to the description in Section A.2.4.[8]

For tuning hyperparameters and reporting test results, we used development and test sets of 2,000 comparisons drawn entirely from the researcher judgments, and fixed across all experiments.

A.3.2 Perplexity

We first compare the Hopkins and May model and TrueSkill using perplexity on the test data T, computed as follows:

$$\text{ppl}(p|T) = 2^{-\sum_{(i,S_1,S_2,\pi)\in T} \log_2 p(\pi|S_1,S_2)} \tag{A.23}$$

where p is the model under consideration. The probability of each observed outcome π between two systems S_1 and S_2 is computed by taking a difference of the Gaussian distri-

[8] We use a Python implementation of TrueSkill (`github.com/sublee/trueskill`).

butions associated with those systems:

$$\begin{aligned}\mathcal{N}(\mu_\delta, \sigma_\delta^2) &= \mathcal{N}(\mu_{S_1}, \sigma_{S_1}^2) - \mathcal{N}(\mu_{S_2}, \sigma_{S_2}^2) \\ &= \mathcal{N}(\mu_{S_1} - \mu_{S_2}, \sigma_{S_1}^2 + \sigma_{S_2}^2) \end{aligned} \tag{A.24}$$

This Gaussian can then be carved into three pieces: the area where S_1 loses, the middle area representing ties (defined by a decision radius, r, whose value is fit using development data), and a third area representing where S_1 wins. By integrating over each of these regions, we have a probability distribution over these outcomes:

$$p(\pi \mid S_1, S_2) = \begin{cases} \int_{-\infty}^{0} \mathcal{N}(\mu_\delta, \sigma_\delta^2) & \text{if } \pi \text{ is } > \\ \int_{0}^{r} \mathcal{N}(\mu_\delta, \sigma_\delta^2) & \text{if } \pi \text{ is } = \\ \int_{r}^{\infty} \mathcal{N}(\mu_\delta, \sigma_\delta^2) & \text{if } \pi \text{ is } < \end{cases} \tag{A.25}$$

We do not compute perplexity for the Expected Wins model, which does not put any probability mass on ties.

A.3.3 Accuracy

Perplexity is often viewed as a neutral metric, but without access to unbounded training data or the true model parameters, it can only be approximated. Furthermore, it does not always correlate perfectly with evaluation metrics. As such, we also present accuracy

results, measuring each model's ability to predict the values of the ternary pairwise judgments made by the annotators. These are computed using the above equation, picking the highest value of $p(\pi)$ for all annotations between each system pair (S_i, S_j). As with perplexity, we emphasize that these predictions are functions of the system pair only, and not the individual sentences under consideration, so the same outcome is always predicted for all sentences between a system pair.

A.3.4 Parameter Tuning

We follow the settings described in Hopkins and May (2013) for their model: $\sigma_a = 0.5, \sigma_{\text{obs}} = 1.0$, and $d = 0.5$. In TrueSkill, in accordance with the Hopkins and May model, we set the initial μ and σ values for each system to 0 and 0.5 respectively, and ϵ to 0.25.

For test data, we tuned the "decision radius" parameter r by doing grid search over $\{0.001, 0.01, 0.1, 0.3, 0.5\}$, searching for the value which minimized perplexity and maximized accuracy on the development set. We do this for each model and language pair. When tuned by perplexity, r is typically either 0.3 or 0.5 for both models and language pairs, whereas, for accuracy, the best r is either 0.001, 0.01, or 0.1.

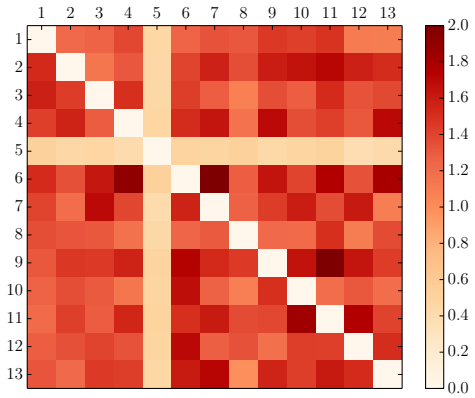

Figure A.3: Heat map for the ratio of pairwise judgments across the full cross-product of systems in the WMT13 French-English translation task.

A.4 Reduced Data Collection with Non-uniform Match Selection

As mentioned earlier, a drawback of the selection of training data for annotation is that it is sampled uniformly from the space of system pair competitions, and an advantage of TrueSkill is its ability to instead compute a distribution over pairings and thereby focus annotation efforts on competitive matches. In this section, we report results in the form of heat maps indicating the percentage of pairwise judgments requested by TrueSkill across the full cross-product of system pairs, using the WMT13 French-English translation task.

Figure A.3 depicts a system-versus-system heat map for all judgments in the dataset. Across this figure and the next two, systems are sorted along each axis by the final values of μ inferred by TrueSkill during training, and the heat of each square is proportional to the percentage of judgments obtained between those two systems. The diagonal reflects

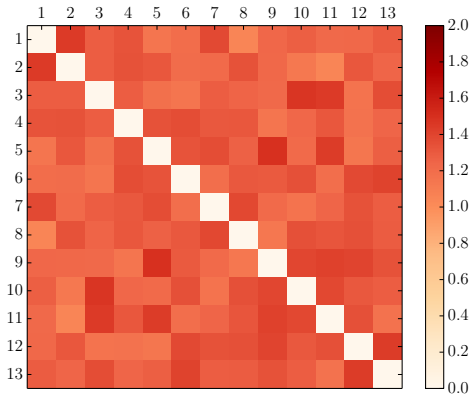

Figure A.4: Heat map for the ratio of pairwise judgments across the full cross-product of systems used in the *first* 20% of TrueSkill model.

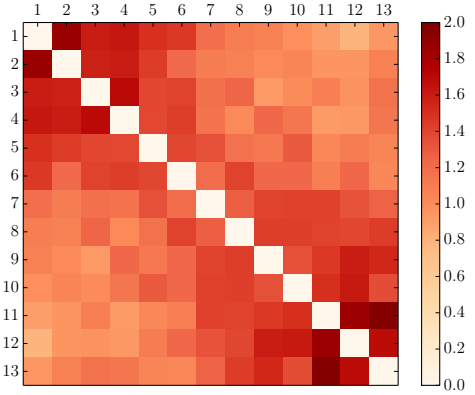

Figure A.5: Heat map for the ratio of pairwise judgments across the full cross-product of systems used in the *last* 20% of TrueSkill model.

the fact that systems do not compete against themselves, and the stripe at row and column 5 reflects a system that was entered late into the WMT13 competition and thus had many fewer judgments. It is clear that these values are roughly uniformly distributed. This figure serves as a sort of baseline, demonstrating the lack of patterns in the data-selection process.

The next two figures focus on the data that TrueSkill itself selected for its use from among all of the available data. Figure A.4 is a second heat map presenting the set of

system pairs selected by TrueSkill for the *first* 20% of its matches chosen during training, while Figure A.5 presents a heat map of the *last* 20%. The contrast is striking: whereas the judgments are roughly uniformly distributed at the beginning, the bulk of the judgments obtained for the last set are clustered along the diagonal, where the most competitive matches lie.

Together with the higher accuracy of TrueSkill, this suggests that it could be used to decrease the amount of data that needs to be collected in future WMT human evaluations by focusing the annotation effort on more closely-matched systems.

A.5 Clustering

As pointed out by Koehn (2012), a ranking presented as a total ordering among systems conceals the closeness of comparable systems. In the WMT13 competition, systems are grouped into clusters, which is equivalent to presenting only a *partial* ordering among the systems. Clusters are constructed using bootstrap resampling to infer many system rankings. From these rankings, *rank ranges* are then collected, which can be used to construct 95% confidence intervals, and, in turn, to cluster systems whose ranges overlap. We use a similar approach for clustering in the TrueSkill model. We obtain rank ranges for each system by running the TrueSkill model 100 times,[9] throwing out the top and bottom 2 rankings for each system, and clustering where rank ranges overlap. For comparison, we also do this for the other two models, altering the amount of training data from 1k to 25k

[9] We also tried the sampling 1,000 times and the clustering granularities were the same.

in increments of 1,000, and plotting the number of clusters that can be obtained from each technique on each amount of training data.

Figure A.6 show the number of clusters according to the increase of training data for three models. TrueSkill efficiently split the systems into clusters compared to other two methods. Figure A.7 and A.8 present the result of clustering two different size of training data (1K and 25K pairwise comparisons) on the TrueSkill model, which indicates that the rank ranges become narrow and generate clusters reasonably as the number of training samples increases. The ranking and clusters are slightly different from the official result (Table A.1) mainly because the official result is based on Expected Wins.

One noteworthy observation is that the ranking of systems between Figure A.7 and Figure A.8 is the same, further corroborating the stability and accuracy of the TrueSkill model even with a small amount of data. Furthermore, while the need to cluster systems forces the collection of significantly more data than if we wanted only to report a total ordering, TrueSkill here produces nicely-sized clusters with only 25K pairwise comparisons, which is nearly one-third large of that used in the WMT13 campaign (80K for French-English, yielding 8 clusters).

A.6 Conclusion

Models of "relative ability" (Koehn, 2012; Hopkins and May, 2013) are a welcome addition to methods for inferring system rankings from human judgments. The TrueSkill

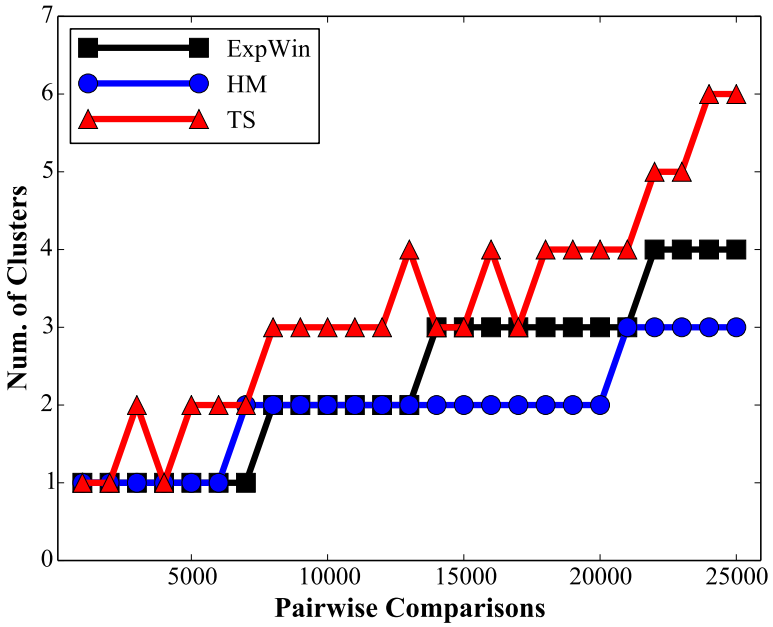

Figure A.6: The number of clusters according to the increase of training data for WMT13 French-English (13 systems in total).

variant presented in this paper is a promising further development, both in its ability to achieve higher accuracy levels than alternatives, and in its ability to sample non-uniformly from the space of system pair matchings. It's possible that future WMT evaluations could significantly reduce the amount of data they need to collect, also potentially allowing them to draw from expert annotators alone (the developers of the participating systems), without the need to hire non-experts on Mechanical Turk.

One piece missing from the methods explored and proposed in this paper is models of the actual translations being compared by judges. Clearly, it is properties of the sentences themselves that judges use to make their judgments, a fact which is captured only indirectly by modeling translation qualities sampled from system abilities. This observation has been

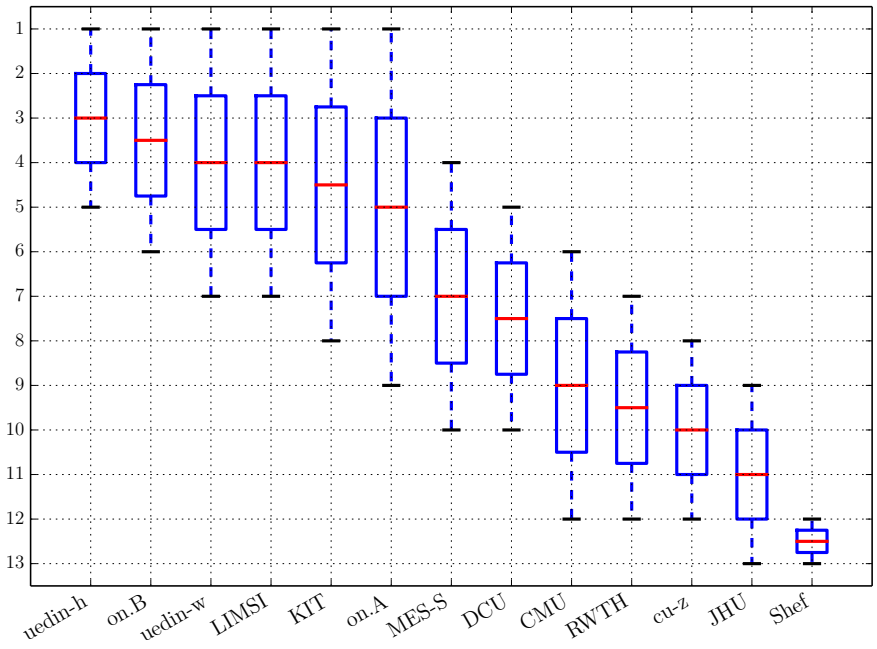

Figure A.7: The result of clustering by TrueSkill model with 1K training data from WMT13 French-English. The boxes range from the lower to upper quartile values, with means in the middle. The whiskers show the full range of each system's rank after the bootstrap resampling.

used in the development of automatic evaluation metrics (Song and Cohn, 2011), and is something we hope to explore in future work for system ranking.

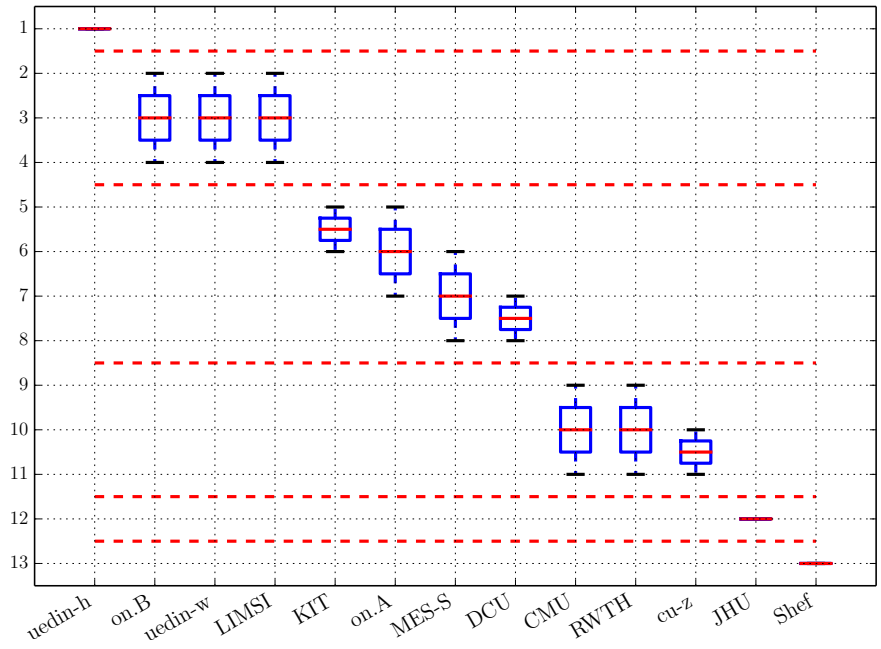

Figure A.8: The result of clustering by TrueSkill model with 25K training data. Dashed lines separate systems with non-overlapping rank ranges, splitting the data into clusters.

Bibliography

Ando, Rie Kubota and Tong Zhang (2005). "A framework for learning predictive structures from multiple tasks and unlabeled data". In: *Journal of Machine Learning Research* 6.Nov, pp. 1817–1853.

Asano, Hiroki, Tomoya Mizumoto, and Kentaro Inui (2017). "Reference-based Metrics can be Replaced with Reference-less Metrics in Evaluating Grammatical Error Correction Systems". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 343–348.

Attali, Yigal and Jill Burstein (2006). "Automated essay scoring with e-rater® V. 2". In: *The Journal of Technology, Learning and Assessment* 4.3.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural machine translation by jointly learning to align and translate". In: *arXiv:1409.0473*.

Ballesteros, Miguel, Chris Dyer, and Noah A. Smith (2015). "Improved transition-based parsing by modeling characters instead of words with LSTMs". In: *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, pp. 349–359.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2, pp. 157–166.

Berzak, Yevgeni, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza, and Boris Katz (2016). "Universal Dependencies for Learner English". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 737–746.

Bojar, Ondrej, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna (2014). "Findings of the 2014 Workshop on Statistical Machine Translation". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, pp. 12–58.

Bojar, Ondřej, Miloš Ercegovčević, Martin Popel, and Omar Zaidan (2011). "A Grain of Salt for the WMT Manual Evaluation". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 1–11.

BIBLIOGRAPHY

Bojar, Ondřej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia (2013). "Findings of the 2013 Workshop on Statistical Machine Translation". In: *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1–44.

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi (2015). "Findings of the 2015 Workshop on Statistical Machine Translation". In: *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1–46.

Brockett, Chris, William B Dolan, and Michael Gamon (2006). "Correcting ESL errors using phrasal SMT techniques". In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 249–256.

Bryant, Christopher and Hwee Tou Ng (2015). "How Far are We from Fully Automatic High Quality Grammatical Error Correction?" In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 697–707.

BIBLIOGRAPHY

Burstein, Jill and Martin Chodorow (1999). "Automated essay scoring for nonnative English speakers". In: *Proceedings of a Symposium on Computer Mediated Language Assessment and Evaluation in Natural Language Processing*. Association for Computational Linguistics, pp. 68–75.

Cahill, Aoife (2015). "Parsing Learner Text: to Shoehorn or not to Shoehorn". In: *Proceedings of The 9th Linguistic Annotation Workshop*. Denver, Colorado, USA: Association for Computational Linguistics, pp. 144–147.

Cahill, Aoife, Nitin Madnani, Joel Tetreault, and Diane Napolitano (2013). "Robust Systems for Preposition Error Correction Using Wikipedia Revisions". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 507–517.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Omar Zaidan (2011). "Findings of the 2011 Workshop on Statistical Machine Translation". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 22–64.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia (2012). "Findings of the 2012 Workshop on Statistical Machine Translation". In: *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Montréal, Canada: Association for Computational Linguistics, pp. 10–51.

BIBLIOGRAPHY

Chodorow, Martin and Claudia Leacock (2000). "An Unsupervised Method for Detecting Grammatical Errors". In: *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pp. 140–147.

Chodorow, Martin, Markus Dickinson, Ross Israel, and Joel Tetreault (2012). "Problems in Evaluating Grammatical Error Detection Systems". In: *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 611–628.

Chollampatt, Shamil, Duc Tam Hoang, and Hwee Tou Ng (2016). "Adapting Grammatical Error Correction Based on the Native Language of Writers with Neural Network Joint Models". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1901–1911.

Chollampatt, Shamil, Kaveh Taghipour, and Hwee Tou Ng (2016). "Neural network translation models for grammatical error correction". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, pp. 2768–2774.

Chomsky, Noam (1957). *Syntactic structure*. Mouton.

Choshen, Leshem and Omri Abend (2018a). "Automatic Metric Validation for Grammatical Error Correction". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 1372–1382.

Choshen, Leshem and Omri Abend (2018b). "Reference-less Measure of Faithfulness for Grammatical Error Correction". In: *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans, Louisiana: Association for Computational Linguistics, pp. 124–129.

Chrupala, Grzegorz (2013). "Text segmentation with character-level text embeddings". In: *arXiv preprint arXiv:1309.4628*.

Chrupała, Grzegorz (2014). "Normalizing tweets with edit scripts and recurrent neural embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 680–686.

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv:1412.3555*.

Church, Kenneth, Ted Hart, and Jianfeng Gao (2007). "Compressing Trigram Language Models With Golomb Coding". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 199–207.

Collins, Michael (2002). "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1–8.

BIBLIOGRAPHY

Dahlmeier, Daniel and Hwee Tou Ng (2011). "Grammatical Error Correction with Alternating Structure Optimization". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 915–923.

Dahlmeier, Daniel and Hwee Tou Ng (2012a). "A Beam-Search Decoder for Grammatical Error Correction". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, pp. 568–578.

Dahlmeier, Daniel and Hwee Tou Ng (2012b). "Better Evaluation for Grammatical Error Correction". In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 568–572.

Dahlmeier, Daniel, Hwee Tou Ng, and Siew Mei Wu (2013). "Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English". In: *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 22–31.

Dale, Robert, Ilya Anisimoff, and George Narroway (2012). "HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task". In: *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 54–62.

BIBLIOGRAPHY

Dale, Robert and Adam Kilgarriff (2011). "Helping Our Own: The HOO 2011 Pilot Shared Task". In: *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*. Nancy, France: Association for Computational Linguistics, pp. 242–249.

Davis, Matt (2003). "Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy". In: *http://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/*.

Devlin, Jacob, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul (2014). "Fast and Robust Neural Network Joint Models for Statistical Machine Translation". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 1370–1380.

Favre, Benoit, Bernd Bohnet, and Dilek Hakkani-Tür (2010). "Evaluation of semantic role labeling and dependency parsing of automatic speech recognition output". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5342–5345.

Felice, Mariano and Ted Briscoe (2015). "Towards a standard evaluation method for grammatical error detection and correction". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, pp. 578–587.

BIBLIOGRAPHY

Felice, Mariano, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar (2014). "Grammatical error correction using hybrid systems and type filtering". In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 15–24.

Felice, Rachele De and Stephen G. Pulman (2008). "A classifier-based approach to preposition and determiner error correction in L2 English". In: *Proceedings of COLING*. Manchester, UK.

Forster, Kenneth I, C Davis, C Schoknecht, and R Carter (1987). "Masked priming with graphemically related forms: Repetition or partial activation?" In: *The Quarterly Journal of Experimental Psychology* 39.2, pp. 211–251.

Foster, Jennifer (2007). "Treebanks gone bad". In: *International Journal of Document Analysis and Recognition (IJDAR)* 10.3, pp. 129–145.

Foster, Jennifer and Oistein Andersen (2009). "GenERRate: Generating Errors for Use in Grammatical Error Detection". In: *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*. Boulder, Colorado: Association for Computational Linguistics, pp. 82–90.

Gamon, Michael, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B Dolan, Dmitriy Belenko, and Lucy Vanderwende (2008). "Using contextual speller techniques and language modeling for ESL error correction". In: *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*.

BIBLIOGRAPHY

Gamon, Michael, Claudia Leacock, Chris Brockett, William B Dolan, Jianfeng Gao, Dmitriy Belenko, and Alexandre Klementiev (2009). "Using statistical techniques and web search to correct ESL errors". In: *Calico Journal* 26.3, pp. 491–511.

Goldberg, Yoav and Michael Elhadad (2010). "An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 742–750.

Goldberg, Yoav and Joakim Nivre (2013). "Training Deterministic Parsers with Non-Deterministic Oracles". In: *Transactions of the Association for Computational Linguistics* 1, pp. 403–414.

Grainger, Jonathan, Jean-Pierre Granier, Fernand Farioli, Eva Van Assche, and Walter JB van Heuven (2006). "Letter position information and printed word perception: the relative-position priming constraint." In: *Journal of Experimental Psychology: Human Perception and Performance* 32.4, p. 865.

Grundkiewicz, Roman, Marcin Junczys-Dowmunt, and Edward Gillian (2015). "Human Evaluation of Grammatical Error Correction Systems". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 461–470.

Guerrera, Christine and Kenneth Forster (2008). "Masked form priming with extreme transposition". In: *Language and Cognitive Processes* 23.1, pp. 117–142.

BIBLIOGRAPHY

Guo, Yan and Gulbahar H Beckett (2007). "The hegemony of English as a global language: Reclaiming local knowledge and culture in China". In: *Convergence* 40.1/2, p. 117.

Ha, T.-L., J. Niehues, and A. Waibel (2015). "Lexical Translation Model Using a Deep Neural Network Architecture". In: *ArXiv e-prints*.

Heafield, Kenneth (2011). "KenLM: Faster and Smaller Language Model Queries". In: *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland, United Kingdom, pp. 187–197.

Heilman, Michael, Aoife Cahill, Nitin Madnani, Melissa Lopez, Matthew Mulholland, and Joel Tetreault (2014). "Predicting Grammaticality on an Ordinal Scale". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 174–180.

Herbrich, Ralf, Tom Minka, and Thore Graepel (2006). "TrueSkill™: A Bayesian Skill Rating System". In: *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*. Vancouver, British Columbia, Canada: MIT Press, pp. 569–576.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Honnibal, Matthew and Mark Johnson (2014). "Joint Incremental Disfluency Detection and Dependency Parsing". In: *Transactions of the Association for Computational Linguistics* 2, pp. 131–142.

BIBLIOGRAPHY

Hopkins, Mark and Jonathan May (2013). "Models of Translation Competitions". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1416–1424.

Hovy, Eduard, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel (2006). "OntoNotes: the 90% solution". In: *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, pp. 57–60.

Humphreys, Glyn W, Lindsay J Evett, and Philip T Quinlan (1990). "Orthographic processing in visual word identification". In: *Cognitive Psychology* 22.4, pp. 517 –560.

Izumi, Emi, Kiyotaka Uchimoto, and Hitoshi Isahara (2004). "The Overview of the SST Speech Corpus of Japanese Learner English and Evaluation Through the Experiment on Automatic Detection of Learners' Errors". In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*.

Johnson, Rebecca L, Manuel Perea, and Keith Rayner (2007). "Transposed-letter effects in reading: Evidence from eye movements and parafoveal preview." In: *Journal of Experimental Psychology: Human Perception and Performance* 33.1, p. 209.

Junczys-Dowmunt, Marcin and Roman Grundkiewicz (2014). "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. Baltimore, Maryland: Association for Computational Linguistics, pp. 25–33.

Junczys-Dowmunt, Marcin and Roman Grundkiewicz (2016). "Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1546–1556.

Junczys-Dowmunt, Marcin, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield (2018). "Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 595–606.

Kaji, Nobuhiro and Masaru Kitsuregawa (2014). "Accurate Word Segmentation and POS Tagging for Japanese Microblogs: Corpus Annotation and Joint Modeling with Lexical Normalization". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 99–109.

Kim, Yoon, Yacine Jernite, David Sontag, and Alexander M. Rush (2015). "Character-aware neural language models". In: *arXiv preprint arXiv:1508.06615*.

Koehn, Philipp (2009). *Statistical machine translation*. Cambridge University Press.

BIBLIOGRAPHY

Koehn, Philipp (2012). “Simulating Human Judgment in Machine Translation Evaluation Campaigns”. In: *Proceedings of the 9th International Workshop on Spoken Language Translation (IWSLT)*. Hong Kong, China: International Speech Communication Association, pp. 179–184.

Leacock, Claudia, Martin Chodorow, Michael Gamon, and Joel Tetreault (2014). “Automated grammatical error detection for language learners”. In: *Synthesis lectures on human language technologies* 7.1, pp. 1–170.

Lee, John (2004). “Automatic Article Restoration”. In: *HLT-NAACL 2004: Student Research Workshop*. Ed. by Daniel Marcu Susan Dumais and Salim Roukos. Boston, Massachusetts, USA: Association for Computational Linguistics, pp. 31–36.

Lin, Chin-Yew (2004). “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Ed. by Stan Szpakowicz Marie-Francine Moens. Barcelona, Spain: Association for Computational Linguistics, pp. 74–81.

Ling, Wang, Isabel Trancoso, Chris Dyer, and Alan W. Black (2015). “Character-based neural machine translation”. In: *arXiv preprint arXiv:1511.04586*.

Lopez, Adam (2012). “Putting Human Assessments of Machine Translation Systems in Order”. In: *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Montréal, Canada: Association for Computational Linguistics, pp. 1–9.

Lyons, John (1968). *Introduction to theoretical linguistics*. Cambridge university press.

BIBLIOGRAPHY

Madnani, Nitin, Martin Chodorow, Joel Tetreault, and Alla Rozovskaya (2011). "They Can Help: Using Crowdsourcing to Improve the Evaluation of Grammatical Error Detection Systems". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 508–513.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger (1994). "The Penn Treebank: annotating predicate argument structure". In: *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pp. 114–119.

Marcus, Mitchell P, Mary Ann Marcinkiewicz, and Beatrice Santorini (1993). "Building a large annotated corpus of English: The Penn Treebank". In: *Computational linguistics* 19.2, pp. 313–330.

Mikolov, Tomas, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model". In: *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pp. 1045–1048.

Nagata, Ryo and Keisuke Sakaguchi (2016). "Phrase Structure Annotation and Parsing for Learner English". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1837–1847.

BIBLIOGRAPHY

Nagata, Ryo, Atsuo Kawai, Koichiro Morihiro, and Naoki Isu (2006). "Reinforcing English Countability Prediction with One Countability per Discourse Property". In: *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. Sydney, Australia: Association for Computational Linguistics, pp. 595–602.

Napoles, Courtney, Matthew Gormley, and Benjamin Van Durme (2012). "Annotated gigaword". In: *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, pp. 95–100.

Napoles, Courtney, Keisuke Sakaguchi, and Joel Tetreault (2016). "There's No Comparison: Reference-less Evaluation Metrics in Grammatical Error Correction". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 2109–2115.

Napoles, Courtney, Keisuke Sakaguchi, and Joel Tetreault (2017). "JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: Association for Computational Linguistics, pp. 229–234.

Napoles, Courtney, Keisuke Sakaguchi, Matt Post, and Joel Tetreault (2015). "Ground Truth for Grammatical Error Correction Metrics". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, pp. 588–593.

BIBLIOGRAPHY

Ng, Hwee Tou, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault (2013). "The CoNLL-2013 Shared Task on Grammatical Error Correction". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1–12.

Ng, Hwee Tou, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant (2014). "The CoNLL-2014 Shared Task on Grammatical Error Correction". In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 1–14.

Nicholls, Diane (2003). "The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT". In: *Proceedings of the Corpus Linguistics 2003 conference*, pp. 572–581.

Nivre, Joakim (2004). "Incrementality in Deterministic Dependency Parsing". In: *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. IncrementParsing '04. Barcelona, Spain: Association for Computational Linguistics, pp. 50–57.

Nivre, Joakim (2009). "Non-Projective Dependency Parsing in Expected Linear Time". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, pp. 351–359.

BIBLIOGRAPHY

Novikova, Jekaterina, Ondrej Dušek, and Verena Rieser (2017). "The E2E Dataset: New Challenges for End-to-End Generation". In: *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. arXiv:1706.09254. Saarbrücken, Germany.

Och, Franz Josef (2003). "Minimum Error Rate Training in Statistical Machine Translation". In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan: Association for Computational Linguistics, pp. 160–167.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "BLEU: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 311–318.

Park, Y. Albert and Roger Levy (2011). "Automated Whole Sentence Grammar Correction Using a Noisy Channel Model". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 934–944.

Pavlick, Ellie, Rui Yan, and Chris Callison-Burch (2014). "Crowdsourcing for grammatical error correction". In: *Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, pp. 209–212.

BIBLIOGRAPHY

Perea, Manuel and Stephen J Lupker (2004). "Can CANISO activate CASINO? Transposed-letter similarity effects with nonadjacent letter positions". In: *Journal of Memory and Language* 51.2, pp. 231 –246.

Peressotti, Francesca and Jonathan Grainger (1999). "The role of letter identity and letter position in orthographic priming". In: *Perception & Psychophysics* 61.4, pp. 691–706.

Petrov, Slav and Ryan McDonald (2012). "Overview of the 2012 Shared Task on Parsing the Web". In: *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Qian, Tao, Yue Zhang, Meishan Zhang, Yafeng Ren, and Donghong Ji (2015). "A Transition-based Model for Joint Segmentation, POS-tagging and Normalization". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1837–1846.

Ranzato, Marc'Aurelio, Sumit Chopra, Michael Auli, and Wojciech Zaremba (2015). "Sequence level training with recurrent neural networks". In: *arXiv:1511.06732*.

Rasooli, Mohammad Sadegh and Joel Tetreault (2013). "Joint Parsing and Disfluency Detection in Linear Time". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 124–129.

Rasooli, Mohammad Sadegh and Joel Tetreault (2014). "Non-Monotonic Parsing of Fluent Umm I mean Disfluent Sentences". In: *Proceedings of the 14th Conference of the*

European Chapter of the Association for Computational Linguistics, volume 2: Short Papers. Gothenburg, Sweden: Association for Computational Linguistics, pp. 48–53.

Rayner, Keith, Sarah J. White, Rebecca L. Johnson, and Simon P. Liversedge (2006). "Raeding Wrods With Jubmled Lettres: There Is a Cost". In: *Psychological Science* 17.3, pp. 192–193.

Rei, Marek and Helen Yannakoudakis (2016). "Compositional Sequence Labeling Models for Error Detection in Learner Writing". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1181–1191.

Roark, Brian, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy Kahn, Yang Liu, Mari Ostendorf, John Hale, Anna Krasnyanskaya, Matthew Lease, Izhak Shafran, Matthew Snover, Robin Stewart, and Lisa Yung (2006). "SParseval: Evaluation Metrics for Parsing Speech". In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. Genoa, Italy: European Language Resources Association (ELRA).

Rozovskaya, Alla and Dan Roth (2010). "Annotating ESL Errors: Challenges and Rewards". In: *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*. Los Angeles, California: Association for Computational Linguistics, pp. 28–36.

Rozovskaya, Alla and Dan Roth (2011). "Algorithm Selection and Model Adaptation for ESL Correction Tasks". In: *Proceedings of the 49th Annual Meeting of the Associa-*

tion for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, pp. 924–933.

Rozovskaya, Alla and Dan Roth (2014). "Building a State-of-the-Art Grammatical Error Correction System". In: *Transactions of the Association for Computational Linguistics* 2, pp. 414–434.

Rozovskaya, Alla, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash (2014). "The Illinois-Columbia System in the CoNLL-2014 Shared Task". In: *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 34–42.

Sakaguchi, Keisuke, Courtney Napoles, and Joel Tetreault (2017). "GEC into the future: Where are we going and how do we get there?" In: *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 180–187.

Sakaguchi, Keisuke, Matt Post, and Benjamin Van Durme (2014). "Efficient Elicitation of Annotations for Human Evaluation of Machine Translation". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, pp. 1–11.

Sakaguchi, Keisuke, Matt Post, and Benjamin Van Durme (2017a). "Error-repair Dependency Parsing for Ungrammatical Texts". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 189–195.

BIBLIOGRAPHY

Sakaguchi, Keisuke, Matt Post, and Benjamin Van Durme (2017b). "Grammatical Error Correction with Neural Reinforcement Learning". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, pp. 366–372.

Sakaguchi, Keisuke, Tomoya Mizumoto, Mamoru Komachi, and Yuji Matsumoto (2012). "Joint English Spelling Error Correction and POS Tagging for Language Learners Writing". In: *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 2357–2374.

Sakaguchi, Keisuke, Courtney Napoles, Matt Post, and Joel Tetreault (2016). "Reassessing the Goals of Grammatical Error Correction: Fluency Instead of Grammaticality". In: *Transactions of the Association for Computational Linguistics* 4, pp. 169–182.

Sakaguchi, Keisuke, Kevin Duh, Matt Post, and Benjamin Van Durme (2017). "Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Pp. 3281–3287.

Schmaltz, Allen, Yoon Kim, Alexander M. Rush, and Stuart Shieber (2016). "Sentence-level grammatical error identification as sequence-to-sequence correction". In: *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. San Diego, CA: Association for Computational Linguistics, pp. 242–251.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone,

Jozef Mokry, and Maria Nadejde (2017). “Nematus: a Toolkit for Neural Machine Translation”. In: *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: Association for Computational Linguistics, pp. 65–68.

Shen, Shiqi, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu (2016). “Minimum Risk Training for Neural Machine Translation”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany: Association for Computational Linguistics, pp. 1683–1692.

Song, Xingyi and Trevor Cohn (2011). “Regression and Ranking based Optimisation for Sentence Level MT Evaluation”. In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 123–129.

Sutskever, Ilya, James Martens, and Geoffrey E Hinton (2011). “Generating text with recurrent neural networks”. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*, pp. 3104–3112.

Sutton, Richard S, David A McAllester, Satinder P Singh, Yishay Mansour, et al. (1999). “Policy gradient methods for reinforcement learning with function approximation.” In: *NIPS*. Vol. 99, pp. 1057–1063.

BIBLIOGRAPHY

Swanson, Ben and Elif Yamangil (2012). "Correction Detection and Error Type Selection as an ESL Educational Aid". In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, pp. 357–361.

Tajiri, Toshikazu, Mamoru Komachi, and Yuji Matsumoto (2012). "Tense and Aspect Error Correction for ESL Learners Using Global Context". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Jeju Island, Korea: Association for Computational Linguistics, pp. 198–202.

Tetreault, Joel and Martin Chodorow (2008). "Native Judgments of Non-Native Usage: Experiments in Preposition Error Detection". In: *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics*. Manchester, UK: Coling 2008 Organizing Committee, pp. 24–32.

Tetreault, Joel, Martin Chodorow, and Nitin Madnani (2014). "Bucking the trend: Improved evaluation and annotation practices for ESL error detection systems". In: *Language Resources and Evaluation* 48.1, pp. 5–31.

Tetreault, Joel, Elena Filatova, and Martin Chodorow (2010). "Rethinking Grammatical Error Annotation and Evaluation with the Amazon Mechanical Turk". In: *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*. Los Angeles, California: Association for Computational Linguistics, pp. 45–48.

BIBLIOGRAPHY

Van Assche, Eva and Jonathan Grainger (2006). "A study of relative-position priming with superset primes." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 32.2, p. 399.

Williams, Ronald J (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4, pp. 229–256.

Wolfe, Travis, Annabelle Carrell, Mark Dredze, and Benjamin Van Durme (2018). "Summarizing Entities using Distantly Supervised Information Extractors." In: *ProfS/KG4IR/Data: Search@ SIGIR*, pp. 51–58.

Wu, Shuangzhi, Dongdong Zhang, Ming Zhou, and Tiejun Zhao (2015). "Efficient Disfluency Detection with Transition-based Parsing". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 495–503.

Xue, Huichao and Rebecca Hwa (2014). "Improved Correction Detection in Revised ESL Sentences". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 599–604.

Yamada, Hiroyasu and Yuji Matsumoto (2003). "Statistical Dependency Analysis with Support Vector Machines". In: *In Proceedings of IWPT*, pp. 195–206.

Yannakoudakis, Helen, Ted Briscoe, and Ben Medlock (2011). "A New Dataset and Method for Automatically Grading ESOL Texts". In: *Proceedings of the 49th Annual Meeting of*

the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon: Association for Computational Linguistics, pp. 180–189.

Yoshikawa, Masashi, Hiroyuki Shindo, and Yuji Matsumoto (2016). "Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, pp. 1036–1041.

Yuan, Zheng and Ted Briscoe (2016). "Grammatical error correction using neural machine translation". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 380–386.

Vita

Keisuke Sakaguchi was born in Tokyo, Japan, and graduated Waseda University, receiving a BA in Literature (major in Philosophy) in 2005. He graduated with MA in Psycholinguistics and Neurolinguistics from University of Essex in 2006. After several years of working at RIKEN Brain Science Institute and IBM Systems Engineering Japan, he started his research on Natural Language Processing at Nara Institute of Science and Technology, receiving ME in Information Science in 2011. In 2013, he entered the Ph.D. program in computer science at Johns Hopkins University, where he studied under Benjamin Van Durme and Matt Post. He interned at Educational Testing Service (ETS) in 2014, and at IBM T.J. Watson Research Center in 2017. He received an Outstanding Paper Award at ACL 2017.