Technical Report

UCAM-CL-TR-904
ISSN 1476-2986

Number 904

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Grammatical error correction in non-native English

Zheng Yuan

March 2017

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted September 2016 by the author for the degree of Doctor of Philosophy to the University of Cambridge, St. Edmund's College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Grammatical error correction in non-native English

Zheng Yuan

Grammatical error correction (GEC) is the task of automatically correcting grammatical errors in written text. Previous research has mainly focussed on individual error types and current commercial proofreading tools only target limited error types. As sentences produced by learners may contain multiple errors of different types, a practical error correction system should be able to detect and correct all errors.

In this thesis, we investigate GEC for learners of English as a Second Language (ESL). Specifically, we treat GEC as a translation task from incorrect into correct English, explore new models for developing end-to-end GEC systems for all error types, study system performance for each error type, and examine model generalisation to different corpora. First, we apply Statistical Machine Translation (SMT) to GEC and prove that it can form the basis of a competitive all-errors GEC system. We implement an SMT-based GEC system which contributes to our winning system submitted to a shared task in 2014. Next, we propose a ranking model to re-rank correction candidates generated by an SMT-based GEC system. This model introduces new linguistic information and we show that it improves correction quality. Finally, we present the first study using Neural Machine Translation (NMT) for GEC. We demonstrate that NMT can be successfully applied to GEC and help capture new errors missed by an SMT-based GEC system.

While we focus on GEC for English, our methods presented in this thesis can be easily applied to any language.

Acknowledgements

First and foremost, I owe a huge debt of gratitude to my supervisor, Ted Briscoe, who has patiently guided me through my PhD and always been very helpful, understanding and supportive. I cannot thank him enough for providing me with opportunities that helped me grow as a researcher and a critical thinker.

I am immensely grateful to my examiners, Paula Buttery and Stephen Pulman, for their thorough reading of my thesis, their valuable comments and an enjoyable viva. My appreciation extends to my fellow members of the Natural Language and Information Processing research group, with whom I have always enjoyed discussing our work and other random things. My gratitude goes to Stephen Clark and Ann Copestake for giving me early feedback on my work as well as Christopher Bryant for generously reading my thesis draft. I would especially like to thank Mariano Felice for being not just a great colleague but also a dear friend. A special mention has to be given to Matthew Purver, who got me interested in this field in the first place.

I am thankful to Cambridge Trust and China Scholarship Council for funding my research, making it possible for me to pursue a doctoral degree in Cambridge. I am also grateful to the Computer Laboratory and St. Edmund's College for supporting my conference attendance.

Finally, I would like to express my heartfelt thanks to my family and friends. Special thanks go to Hui Xiao and Mo Jia for always being there whenever I need them. Words are powerless to describe my appreciation and gratitude to my parents, Xuewen Yuan and Yun Zeng, for all the sacrifices that they have made on my behalf. Their love and support have sustained me thus far and I know will continue to sustain me.

Contents

List of Abbreviations

ALM	adaptive language model
BiRNN	Bidirectional Recurrent Neural Network
BLEU	Bilingual Evaluation Understudy
BNC	British National Corpus
CE	correct edit
CLC	Cambridge Learner Corpus
CNN	Convolutional Neural Network
CoNLL	Conference on Computational Natural Language Learning
EM	Expectation-Maximisation
ESL	English as a Second Language
EVP	English Vocabulary Profile
FCE	First Certificate in English
FN	false negative
FP	false positive
GEC	grammatical error correction
GLEU	Generalized Language Evaluation Understanding
GPU	graphics processing unit
GR	grammatical relation
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HOO	Helping Our Own
I	I-measure
IELTS	International English Language Testing System
ILP	Integer Linear Programming
ITG	Inversion Transduction Grammar
L1	first language
L2	second language
LM	language model
LSTM	Long Short-Term Memory
MBR	Minimum Bayes-Risk
ME	missed edit
MEMT	Multi-Engine Machine Translation
MERT	Minimum Error Rate Tuning
MIRA	Margin Infused Relaxed Algorithm
MT	machine translation

NB	Naïve Bayes
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NMT	Neural Machine Translation
NP	noun phrase
NUCLE	National University of Singapore Corpus of Learner English
NUS	National University of Singapore
OOV	out-of-vocabulary
P	precision
POS	part-of-speech
R	recall
RASP	Robust Accurate Statistical Parsing
RBMT	Rule-Based Machine Translation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMT	Statistical Machine Translation
SVM	Support Vector Machine
TM	translation model
TN	true negative
TP	true positive
UE	unnecessary edit
WAcc	weighted accuracy
WMT	Workshop on Statistical Machine Translation

CHAPTER 1

INTRODUCTION

Today, from Beijing to Brasilia, millions of people are learning English as a Second Language (ESL). According to a report published by the British Council in 2013, English is spoken at a 'useful level' by 1.75 billion people worldwide. In fact, non-native English speakers now outnumber native speakers. Furthermore, the number of ESL learners keeps on growing and it is estimated that 2 billion people will be using English - or learning to use it - by 2020. Nevertheless, learning a new language is never easy. Difficulties in acquiring a new language can be due to the differences between the new language and the learners' first languages (L1s) (Lado, 1957). These differences may result in various kinds of errors in learner writing. Errors made by learners are different from those made by native speakers. Connors and Lunsford (1988) studied errors made by college students in the United States and compiled an error list ranked by frequency. Their work was later replicated by Donahue (2001) with a focus on ESL learners. Results showed that half of the ten most frequent error types made by native speakers were 'negligible' in ESL writings.

There has been a great deal of commercial and academic interest in automatically correcting these written errors for ESL learners. From a commercial perspective, there is a great potential for many practical applications, such as proofreading tools that help second language (L2) speakers identify and correct their writing errors without human intervention or educational software for automated language learning and assessment. From a research perspective, correcting errors in learner writing is an interesting and challenging task as it involves various aspects of Natural Language Processing (NLP), such as language modelling, syntax and semantics.

Early grammar checkers can be traced back to the 1980s, when hand-coded grammar rules were mostly used. However, due to the productive nature of language and the creativity of learners, it is impractical to define rules for every possible case. With the advent of large-scale annotated corpora in the 1990s, data-driven approaches made it possible to build systems for specific error types. Nevertheless, popular commercial proofreading tools only target a few error types that are easy to correct, such as spelling mistakes (*a *baeutiful/beautiful girl*) or wrong past participle forms of irregular verbs (*Dave has *runned/run 42 marathons*), and do not include those aspects of English that are harder to learn. At the same time, most research in the area has focussed on two common error types made by learners, namely articles

(*Mary's sister is */a hairdresser*) and prepositions (*the best places to visit *on/in July*), assuming that there is only one error per sentence.

However, errors do not always occur in isolation. Sentences produced by learners may contain multiple errors which belong to different error types. What is worse, errors may interact with others so that the correction of one error requires the correction of the other. See the following example sentences written by ESL learners:

Example 1.1. *I am plece to tell the information do you need for the group.*

The sentence contains three errors: a spelling mistake (*plece* → *pleased*), a wrong verb (*tell* → *provide*) and an unecessary verb (*do*).

Example 1.2. *As you know, it is not suitable to wear a jean.*

The sentence contains two interacting errors: 'a' should be deleted and 'jean' should be changed to 'jeans' at the same time (*a jean* → *jeans*).

An error correction system that can only correct one or a few types of errors will be of limited use to learners. Instead, a good system should be able to correct a variety of error types and corrections should be performed at a global rather than local level, including taking interacting errors into account. Our goal in this thesis is to develop robust error correction systems that can automatically detect and correct all errors present in learner text, trying to overcome the aforementioned limitations.

The error correction task can be thought of as a type of monolingual 'translation', where the source is a sentence written by a learner and the target is a fluent and adequate sentence in the same language. A corrected sentence should be grammatically correct and preserve the original meaning of the source.

Rather than building individual components for each error type, we apply the machine translation (MT) approach of 'translating' a grammatically incorrect sentence into a correct one to address all error types simultaneously. The MT approach takes advantage of large annotated learner data. Systems learn correction mappings from data and use them to generate a corrected version of the original sentence, correcting as many errors as possible. Our work investigates MT methods for correcting grammatical errors in non-native English text and addresses issues arising from applying existing MT techniques to the error correction task. We further identify new techniques for developing robust error correction systems that outperform previous approaches.

While English is by far the most spoken foreign language in the world, there is also a need for grammar checkers for other languages, such as Chinese, Spanish and Arabic. Although we focus only on English in this thesis, the methods described here can be applied to any language given appropriate data.

1.1 What is grammatical error correction?

Grammatical error correction (GEC) is the task of automatically correcting grammatical errors in written text. More specifically, the task is to build a system that takes an input text, analyses the context of the text to identify and correct any grammatical errors, and finally returns a corrected version that retains the original meaning. If there is no error in the input text, the system should output the

text without any modification. In this thesis, we focus on grammatical errors in non-native English text.

It should be noted that not all errors present in learner text are grammatical errors, however. Errors were traditionally identified at five levels: 1) a lexical level, 2) a syntactic level, 3) a semantic level, 4) a discourse structure level, and 5) a pragmatic level (Kukich, 1992). Lexical errors are spelling mistakes that result in non-existent words, such as misspelling 'type' as 'tipe', where 'tipe' is not a legitimate word in English. Errors where the syntactic categories of the words do not fit their contexts are classified as syntactic errors, such as subject-verb agreement errors (*she always *know/knows her place*) or verb tense errors (*the church *is/was rebuilt in 1948*). Errors that cause semantic anomalies are semantic errors, which involve contextual spelling mistakes that result in legitimate words (*we waited for twenty *minuets/minutes*) and collocation/cooccurrence errors (**big conversation*) (Kochmar, 2016). Discourse errors violate the inherent coherence relations in a text while pragmatic errors reflect some anomalies related to the goals and plans of the discourse participants. Correcting errors from the last two groups requires further discourse analysis. In this thesis, we use the broad term 'grammatical error' to refer only to lexical, syntactic and semantic errors, but do not tackle discourse and pragmatic errors whose 'span' goes beyond the sentence.

1.2 Thesis aims

The work presented in this thesis aims to:

1. Develop end-to-end error correction systems that are capable of correcting grammatical errors present in text written by learners of English. As sentences produced by ESL learners may contain multiple errors which belong to different error types, we aim to develop robust systems that can automatically detect and correct a variety of error types and perform corrections at a global rather than local level, where interacting errors are covered as well.

2. Explore the use of several statistical NLP approaches for GEC:

   (a) **Can SMT form the basis of a competitive all-errors GEC system?** Statistical Machine Translation (SMT) has been successfully used to correct a limited number of grammatical errors in the past (see Brockett et al., 2006; Yuan and Felice, 2013), so we aim to investigate whether the same approach can be used to correct multiple grammatical errors at the same time.

   (b) **Can candidate re-ranking improve sentence quality in SMT-based GEC?** Since SMT was not originally designed for GEC, many standard features do not perform well on the error correction task. It is therefore necessary to add new local and global features to help the SMT decoder distinguish good from bad corrections. We propose a Support Vector Machine (SVM) ranking model to re-rank candidates generated

by an SMT-based GEC system. We aim to determine whether candidate re-ranking is a viable approach to address the decoding problem in this scenario and thus improve sentence quality.

(c) **Can NMT be applied to GEC?** Typical SMT-based GEC systems suffer from data sparsity. Some errors are not covered by these systems because the mappings needed for correction have not been seen in the training data. With the recent advances in neural networks, Neural Machine Translation (NMT) seems appealing for GEC as it may be possible to correct erroneous phrases and sentences that have not been seen in the training set more effectively. We investigate NMT systems and how they can be applied to GEC in order to capture new errors without the need for additional training data.

3. Examine and address issues concerning applying existing techniques to GEC. As we approach GEC as a special translation task, where the source and target sentences are both in English but the source may contain grammatical errors, it is inevitable that new problems may arise from adapting existing MT techniques to GEC. We discuss these problems and propose possible solutions.

4. Investigate system performance for each error type. Type-specific performance helps understand the strengths and weaknesses of the system, as well as identify areas for future improvement. However, this is not easy to do for all-errors GEC systems which propose corrections without error types. We apply a type estimation strategy and present detailed error analyses.

5. Examine model generalisation to different learner corpora. It is not the aim of this thesis to beat the state-of-the-art result on one particular dataset (e.g. the CoNLL-2014 shared task test set - see Section 2.5.2). Instead, we are more interested in models that can consistently produce competitive results across different learner corpora without retraining or tuning for new datasets or GEC tasks. For this reason, we test model generalisation and compare the results with those from other models which are trained and tuned specifically for each corpus.

1.3 Thesis structure

The structure of this thesis is as follows. Chapter 2 discusses several related topics in GEC. It begins with an overview of the automated approaches to detect and correct errors made by learners and goes on to describe the MT approach to error correction. Additionally, it gives a description of the learner corpora and automatic evaluation metrics for GEC, followed by a summary of a series of shared tasks on GEC. It concludes with a discussion of the datasets and evaluation metrics used in this thesis.

Chapter 3 describes our approach to building a preliminary SMT-based error correction system. We address the major issues that arise from applying standard SMT to GEC. We explore different types of translation models (TMs), language

models (LMs) and alignment methods used in an SMT system. To overcome the lack of training data, we propose the use of three different types of data and demonstrate how they can help build robust SMT models. We also investigate phrase table filtering. We present an SMT system that forms one half of our winning system submitted to a shared task on grammatical error correction in 2014. A detailed error analysis of the SMT-based GEC system is also performed.

In Chapter 4, we propose a supervised ranking model to re-rank candidates generated by an SMT-based GEC system. A range of novel features with respect to error correction are investigated and implemented in our re-ranker. An in-depth assessment of the role played by each feature type is carried out, quantifying its contribution from a statistical perspective. We also investigate the performance of different re-ranking techniques and find that our proposed model clearly outperforms the other two, showing its effectiveness in re-ranking candidates for GEC.

Chapter 5 presents the first study on NMT for GEC, in an attempt to ameliorate the lack of training data for SMT-based GEC systems. Problems from adapting standard NMT to GEC are addressed. The performance of different NMT models on the error correction task is investigated. We also propose a two-step approach to address the 'rare word' problem in NMT for GEC and demonstrate how it can help provide a substantial improvement in system performance.

Finally, Chapter 6 concludes this thesis and discusses some avenues for possible future research.

CHAPTER 2

Background

There is a large body of work on grammatical error detection and correction. This chapter puts the present research in context by offering an overview of latest research in the field. A more comprehensive survey of automated grammatical error detection for language learners can be found in the book by Leacock et al. (2014).

2.1 Early approaches to grammatical error correction

Early attempts at automated error correction employed hand-coded rules. The first widely used grammar checking tools, such as the *Writer's Workbench* (MacDonald et al., 1982), were based on simple pattern matching and string replacement. Other rule-based systems incorporated syntactic analysis and used manually developed grammar rules. For example, both *Grammatik* from Aspen Software and *GramCheck* (Bustamante and León, 1996) relied on basic linguistic analysis, while *IBM's Epistle* (Heidorn et al., 1982) and *Critique* (Richardson and Braden-Harder, 1988) performed full syntactic analysis. Rule-based approaches are generally easy to implement for some types of errors and can be very effective. This is why they are still widely used by existing grammar checking systems. However, rules can become impractical for some complex errors and unmanageable with time. The highly productive nature of language makes it impossible to define rules for every potential error. So rule-based approaches are often avoided as a general solution.

With the advent of large-scale annotated resources in the 1990s, researchers moved to data-driven approaches and applied machine learning techniques to build classifiers for specific error types (Knight and Chander, 1994; Han et al., 2004; Chodorow et al., 2007; De Felice and Pulman, 2007; Tetreault and Chodorow, 2008; Rozovskaya and Roth, 2011; Dahlmeier et al., 2012). Most work using machine learning classifiers has focussed on two error types: articles and prepositions. This is due to the fact that these errors are some of the most common and challenging ones for ESL learners, and are also easier to tackle using machine learning approaches than hand-crafted rules (Felice and Yuan, 2014a). For these closed-class errors, a

finite confusion set or candidate set including all the possible correction candidates is defined, such as a list of articles or prepositions in English. Training examples - native and/or learner data - are represented as vectors of linguistic features that are considered useful for the error type. Possible features often include neighbouring words, part-of-speech (POS) tags, grammatical relations (GRs) and dependency trees. Various machine learning algorithms are used to train classifiers based on these features. Once a system has been trained, new errors are detected and corrected by comparing the original word used in the text with the most likely candidate predicted by the classifier. Since the most useful features often depend on the word class, it is necessary to build separate classifiers for each error type. Han et al. (2004) trained a maximum entropy classifier to detect article errors on a large diverse corpus and achieved an accuracy of 88%. Tetreault and Chodorow (2008) used maximum entropy models to correct errors for 34 common English prepositions in learner text.

Errors made by ESL learners often depend on their L1s (Lee and Seneff, 2008). Systems perform much better when information about their L1s is included. Rozovskaya and Roth (2011) compared four linear machine learning classifiers for correcting preposition errors. Results showed that discriminative classifiers perform the best and adaptation to a writer's L1 further improves performance. The authors proposed a way of integrating language-specific priors at decision time using Naïve Bayes (NB) models instead of training separate classifiers for each L1.

The weakness of approaches based on 'classification by error type' is that they only rely on local context and treat errors independently, assuming that there is only one error in the context and all the surrounding information is correct. However, sentences produced by learners may contain a complex combination of several types of errors which may further interact. An error correction system that only corrects one type of error is of limited use to language learners in practical applications.

A commonly used solution is to build multiple classifiers and then cascade them into a pipeline system. A combination of classifier-based and rule-based steps is often used to build systems that correct multiple errors (Dahlmeier et al., 2012; Rozovskaya et al., 2013). This kind of solution is complex and laborious: several pre-processing and post-processing steps are required, and the order of classifiers also matters. Additionally, it does not solve the problem of interacting errors and predictions from independent classifiers may be inconsistent. Here is a typical example taken from Rozovskaya and Roth (2013):

Example 2.1. *... electric cars is still regarded as a great trial innovation ...*

Predictions made by a system that combines independently-trained classifiers: *cars is* $\rightarrow$ *car are.*

Several approaches have been proposed to address the problem of interacting errors. Rather than making decisions independently, Dahlmeier and Ng (2012a) developed a *beam-search* decoder to iteratively generate sentence-level candidates and score them using individual classifiers and a general LM. Five *proposers* were used to generate new candidates by making five types of changes: spelling, articles, prepositions, punctuation insertion, and noun number. Results appeared promising and the decoder outperformed a pipeline system of individual classifiers and rule-based steps. However, their decoder only provides corrections for five error types and new

proposers need to be added into the system in order to cover more errors, some of which might not be easy to design. Furthermore, the number of candidates grows exponentially with the type of errors being considered (i.e. the number of *proposers*) and the sentence length. As it is infeasible to enumerate all candidates, building an efficient decoder becomes a problem. Wu and Ng (2013) proposed a joint inference model to resolve inconsistencies produced by individual classifiers. Integer Linear Programming (ILP) was used to incorporate the output of individual classifiers and a list of linguistic constraints. These constraints were manually defined and explicitly encoded into the system. Any new constraints need to be hand-coded for new types of interacting errors. Rozovskaya and Roth (2013) built two joint classifiers to address two linguistic structures: subject-verb and article-NPhead.[1] For each of the structures, rather than using two classifiers independently, a joint classifier simultaneously predicts two words that are part of the same structure. Unlike the ILP model proposed by Wu and Ng (2013), the joint classifier does not need human defined constraints, as it can learn from the training data directly. However, it is more difficult to collect enough pairs of candidates that form the relevant structures to use as training data. As one joint classifier only targets one type of interacting error, new classifiers need to be built for every new type of interaction. These classifier-based approaches still use scores from individual classifiers, so it becomes infinitely time-consuming to train individual classifiers for all types of (interacting) errors.

A more general approach for correcting multiple errors in ESL text is to use n-gram LMs (Gamon et al., 2008; Gamon, 2011). A single model is trained on a large number of correct sentences and then used to assign probabilities to sequences of words based on counts from the training data. Within this framework, the target word sequence is substituted for alternatives from a precompiled candidate set and the LM scores for the original text as well as the alternatives are computed. The sequence with the highest probability is chosen as the correct one. Ideally, correct word sequences will get high probabilities while incorrect or unseen ones will get low probabilities. Errors are assumed to occur in parts of a sentence where a low score is assigned. However, no matter how large a training corpus is, it is impossible to cover all possible correct word sequences in practice. Another problem lies in how to distinguish low-frequency word combinations from erroneous ones. Therefore, the LM approach is commonly used in addition to other approaches, especially to rank correction suggestions proposed by other models. Gamon et al. (2008) used a LM in addition to machine learning classifiers and combined them using a meta-classifier. Dahlmeier and Ng (2012a) used a LM in combination with classifiers to score correction candidates in a *beam-search* decoder.

Additionally, some efforts have been made to tackle learner errors that are particularly difficult to detect and correct. Rozovskaya et al. (2014b) proposed a linguistically motivated approach to verb error correction. Their model integrated a machine learning approach with a rule-based system that first identifies verb candidates in noisy learner text and then makes use of verb finiteness information to identify errors and characterise the type of mistake. Xue and Hwa (2014) developed

[1] article-NPhead: the interaction between the head of the noun phrase (NP) and the article that refers to the NP

a computational model for redundancy detection in ESL writings. They proposed a measure to assign high scores to words and phrases that are likely to be redundant within a given sentence by comparing an ESL sentence with the output from off-the-shelf MT systems. For content word combinations, Kochmar (2016) performed error detection in adjective-noun and verb-object combinations in learner data using compositional distributional semantic models.

2.2 Machine translation and error correction

A practical error correction system should be able to correct various types of errors made by ESL learners. In more recent research, MT techniques have been used to successfully correct a broader set of errors.

MT algorithms automatically translate text from a source language into a target language. Error correction thus can be seen as a special translation problem from grammatically incorrect sentences into correct ones. Unlike in standard MT tasks, the source and target sentences are both in the same language but the source may contain grammatical errors. MT-based GEC systems learn correction mappings from parallel examples and use these mappings to generate a corrected version of the original (erroneous) sentence, correcting as many errors as possible.

2.2.1 Statistical machine translation

SMT, as the dominant MT approach in the last two decades, employs statistical models estimated from parallel corpora (i.e. source-target pairs) and monolingual corpora (i.e. target sentences) to transform text from one language to another.[2] Brockett et al. (2006) first proposed the use of an SMT model for correcting mass/-count noun errors made by learners of English. A list of 14 mass nouns was compiled using dictionaries and the Chinese Learner English Corpus (Gui and Yang, 2003). An SMT system requires millions of examples of correct and incorrect usage to learn reliable translation mappings. Given that examples of correct usage are plentiful in native data while parallel examples of incorrect usage are much more difficult to collect, the authors transformed well-formed edited English sentences into mostly ungrammatical strings by introducing artificial mass noun errors. Hand-constructed regular expressions were used to make sure the generated strings exhibited characteristics of the learner corpus. A phrase-based SMT system was built using word alignments produced by GIZA++ (Och and Ney, 2003). Their SMT system successfully corrected 61.8% of mass noun errors from a set of 123 examples of incorrect usage. As noted by Leacock et al. (2014), this was only a first exploration of SMT techniques for GEC, but with enough training data, such a system could potentially be powerful enough to detect and correct errors that involve more than just the insertion, deletion or substitution of single words, as well as being able to provide stylistic writing assistance to ESL learners.

[2]SMT algorithms are described in more detail in Chapter 3.

Mizumoto et al. (2011) applied the same SMT techniques for Japanese error correction but improved them by considering a wider set of error types and training on a large-scale real-world dataset. Rather than transforming correct sentences into grammatically incorrect strings, they extracted real examples from the language learning social network website Lang-8.[3] Moses (Koehn et al., 2007) was used as a decoder and GIZA++ as an alignment tool. Evaluation was based on a character-level version of the Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002), a popular metric for automatic MT evaluation. Mizumoto et al. (2012) extended their work to English and investigated the effect of training corpus size on various types of grammatical errors. Their results showed that a phrase-based SMT system is effective at correcting errors that can be identified by a local context, but less effective at correcting errors that need long-range contextual information.

Yuan and Felice (2013) trained a POS-factored SMT system to correct five types of errors in learner text for the CoNLL-2013 shared task on grammatical error correction (Ng et al., 2013) (see Section 2.5.2). These five error types involve articles, prepositions, noun number, verb form, and subject-verb agreement. Since the limited in-domain training data was insufficient to train an effective SMT system, we explored alternative ways of generating pairs of incorrect and correct sentences automatically from other existing learner corpora. We also proposed several modifications to address issues that affect system performance, like disabling word reordering and removing incorrect alignment mappings from the phrase table used by the SMT decoder. Although our SMT approach did not yield particularly high performance compared to other teams using machine learning classifiers, nevertheless, this revealed the potential of using SMT as a general approach for correcting multiple error types and interacting errors simultaneously. The version of the corpus used for the shared task only includes five error types and discards all the remaining corrections, resulting in some broken or partly-corrected sentences. These ill-formed sentences are particularly harmful for SMT-based systems which, unlike classifiers, work at a global rather than local level. As a result, many corrections proposed by our SMT system were considered incorrect because they did not belong to any of the five target error types. This showed that the SMT approach seems more suitable for an all-errors task rather than a constrained error correction task.

In the CoNLL-2014 shared task (Ng et al., 2014) (see Section 2.5.2), the top performing systems demonstrated that the SMT framework can yield state-of-the-art performance on an all-errors correction task. Our winning system (Felice et al., 2014) is a pipeline of a rule-based system and a phrase-based SMT system (see Chapter 3). The SMT system was trained on parallel sentences and short phrase alignments extracted from fully annotated learner corpora (see Section 2.3). Word alignment was carried out using Pialign (Neubig et al., 2011). As most words translate into themselves and some errors are often similar to their correct forms, we introduced character-level Levenshtein distance (Levenshtein, 1966), which captures the number of edit operations required to change the source phrase into the target phrase. The 10-best correction candidates produced by the SMT system were then re-ranked using Microsoft's Web N-gram Services, which provide access to large *smoothed* n-

[3] `http://lang-8.com`

gram LMs built from English web documents containing trillions of tokens (Gao et al., 2010). Corrections were finally filtered by error type. Our work showed that an SMT-based GEC system can produce state-of-the-art performance on the task and candidate re-ranking can further improve it.

The SMT framework was also adopted by Junczys-Dowmunt and Grundkiewicz (2014), who ranked third out of the 13 participating teams. Following the work of Mizumoto et al. (2012), they constructed a training corpus of more than 3 million pairs of parallel sentences from Lang-8. Since the Lang-8 data can be quite noisy, they performed error selection by keeping errors that resembled mistakes in a learner corpus and replacing others with their corresponding corrections. Apart from the LM built from the target side of the training data, a 5-gram LM estimated from the entire CommonCrawl data (approximately 440 billion tokens, see Buck et al., 2014) was used during decoding. Similar to our character-level Levenshtein distance feature, they introduced a word-based version. Feature weights were tuned for F-score using the k-best Margin Infused Relaxed Algorithm (MIRA) (Cherry and Foster, 2012) and Minimum Error Rate Tuning (MERT) (Och, 2003). Although they concluded that parameter optimisation was essential, Kunchukuttan et al. (2014) subsequently found that tuning for F-score to increase precision yielded worse performance. Grundkiewicz and Junczys-Dowmunt (2014) later introduced the WikEd Error Corpus, which consists of more than 12 million sentences extracted from Wikipedia revision histories. A similar error selection process was performed to only keep errors that resembled those made by ESL learners.

In a following paper, Junczys-Dowmunt and Grundkiewicz (2016) introduced additional features based on edit operation counts, as well as an operation sequence model (Durrani et al., 2013) and a 9-gram LM based on word-classes produced by *word2vec* (Mikolov et al., 2013). However, the integration of additional models/features seemed to affect the underlying algorithm used in SMT. The authors also observed erratic behaviour when optimising the new features and therefore proposed partial solutions to task-specific parameter tuning. Finally, they reported new state-of-the-art performance on the CoNLL-2014 shared task test set, with an $F_{0.5}$ score of 49.49%.

The 'translation' approach has also been used to perform automatic post-editing. Simard et al. (2007) discussed the use of an SMT system to translate erroneous texts produced by a Rule-Based Machine Translation (RBMT) system into better texts in the same language. A phrase-based SMT system was used as an automatic post-editing system and results showed that the SMT system was effective at correcting repetitive errors made by the RBMT system.

Instead of translating an erroneous English sentence into a correct one directly, an SMT system could be used as an auxiliary tool for producing 'round-trip' translations (Hermet and Désilets, 2009; Madnani et al., 2012). The idea of round-trip SMT is to first translate an English sentence into a *pivot* foreign language, and then translate the *pivot* foreign language sentence back into English. By comparing the original English sentence and the round-trip translation, errors can be detected and corrected. Hermet and Désilets (2009) focussed on sentences containing preposition errors and generated a round-trip translation via French. They simply used the

round-trip translation as the 'correction' for the original sentence and their model was able to correct 66.4% of errors. Madnani et al. (2012) used round-trip translations obtained from the Google Translate API[4] via 8 different *pivot* languages for an all-errors task. Their results showed that it is rarely the case that one *pivot* language could offer a round-trip translation that corrected all errors in the sentence; but that several *pivot* languages, if combined properly, could. An alignment algorithm was designed to combine multiple round-trip translations generated from the API into a lattice using TERp, an extension of the Translation Edit Rate evaluation metric (Snover et al., 2009). The lattice was then used to extract whole-sentence corrections. Their experiments yielded fairly reasonable results but left significant room for improvement.

2.2.2 Candidate re-ranking

Despite the success of SMT-based GEC systems, one of the weaknesses is that SMT features used in the framework might not perform well on the error correction task given that SMT was not originally intended for GEC. Since the SMT features were designed to capture *translation* regularities, they may fail to capture some *correction* regularities. As a result, the correction produced by an SMT system is not always the best. It thus seems necessary to add new features with respect to GEC for building effective SMT-based GEC systems, although work in this direction is very limited.

Felice et al. (2014) and Junczys-Dowmunt and Grundkiewicz (2014) introduced Levenshtein distance to their phrase-based SMT systems. Felice et al. (2014) further used a web-based LM to re-rank the 10-best correction candidates produced by the SMT system.

Re-ranking, on the contrary, has been widely used in many NLP tasks such as parsing, tagging and sentence boundary detection (Collins and Duffy, 2002; Collins and Koo, 2005; Roark et al., 2006; Huang et al., 2007). Various machine learning algorithms have been adapted to these re-ranking tasks, including boosting, perceptrons and SVMs. Over the last decade, re-ranking techniques, especially discriminative re-ranking, have shown significant improvement in MT. For each source sentence, rather than outputting the candidate with the highest probability directly, an n-best list of candidate translations is collected from an SMT system and later re-ranked using re-ranking algorithms.[5] New global and local features that have not been used during translation can then be easily added to the re-ranker, without worrying about fine-grained *smoothing* issues in the SMT framework. Shen et al. (2004) successfully applied discriminative re-ranking to MT and observed an improvement in BLEU over the original output of the SMT system. As phrase-based SMT systems make little or no direct use of syntactic information, Och et al. (2004) proposed to use syntactic features to re-rank the n-best list. A wide range of features were systematically evaluated, including word-level features, shallow syntactic features based on POS tags and chunks, and features from Treebank-based syntactic analyses. However, these syntactic features only gave very small gains and

[4] `https://cloud.google.com/translate`

[5] Re-ranking algorithms are described in more detail in Chapter 4.

improvements were mostly due to the addition of translation probabilities from IBM Models (Brown et al., 1993), a non-syntactic feature. Goh et al. (2010) employed an online training algorithm for SVM-based structured prediction. Various global features were investigated for SMT re-ranking, such as the decoder's scores, source and target sentences, alignments, POS tags, sentence type probabilities, posterior probabilities and back translation features. Farzi and Faili (2015) proposed a re-ranking system based on swarm algorithms, where a set of non-syntactic features that can be easily computed from LMs, TMs, n-best lists of candidates and POS tags were used.

As candidate re-ranking seems potentially valuable for GEC, we propose an SVM ranking model to improve SMT output, making it the first work to use discriminative re-ranking for SMT-based GEC.

2.2.3 Neural machine translation

In the past few years, neural network techniques have found success in a wide range of NLP tasks, such as language modelling (Mnih and Hinton, 2007; Mikolov and Zweig, 2012), discriminative parsing (Collobert, 2011), sentiment analysis (Socher et al., 2011; Glorot et al., 2011) and summarisation (Kågebäck et al., 2014). Thus, it is not surprising that neural network models have also been applied to error detection and correction. Sun et al. (2015), for example, employed a Convolutional Neural Network (CNN) for article error correction. Instead of building machine learning classifiers using pre-defined syntactic and/or semantic features, a CNN model is trained from surrounding words with pre-trained word embeddings. Lee et al. (2016) used a CNN to predict whether a sentence needs editing. Rei and Yannakoudakis (2016) looked into various neural network sequence labelling models for error detection in learner writing.

The tide of neural models has also spread to the field of MT. Unlike SMT, NMT learns a single large neural network which inputs a source sentence and outputs a translation. The use of NMT models has shown promising results for several MT tasks (see Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). Specifically, NMT systems ranked on par with phrase-based SMT systems on a couple of language pairs in the 2015 Workshop on Statistical Machine Translation (WMT) shared translation task (Bojar et al., 2015).[6]

NMT applies an *encoder-decoder* framework. An encoder first reads and encodes an input sentence into a vector representation. A decoder then outputs a translation for the input sentence from the vector representation.[7] Different network architectures have been proposed for NMT. Kalchbrenner and Blunsom (2013) first used a CNN to encode source sentences and a Recurrent Neural Network (RNN) to generate target translations. A similar CNN encoder was then used by Meng et al. (2015). Sutskever et al. (2014) and Cho et al. (2014) used RNNs for both encoding and decoding. Sutskever et al. (2014) used a multilayer Long Short-Term Memory (LSTM) to map a source sentence into a fixed-sized vector, and another LSTM to

[6]The NMT system from Jean et al. (2015b) ranked *1st* on English-German translation task and *3rd* on Czech-English, English-Czech and German-English translation tasks (ties were allowed).

[7]NMT algorithms are described in more detail in Chapter 5.

decode a target sentence from the vector. Cho et al. (2014) used two Gated Recurrent Unit (GRU) models, one as the encoder and another as the decoder. Bahdanau et al. (2015) introduced an attention mechanism to NMT which helps the decoder focus on the most relevant information in a source sentence when predicting target words. Luong et al. (2015a) experimented with two attention mechanisms and compared various alignment functions. Both Bahdanau et al. (2015) and Luong et al. (2015a) have shown that attention-based models are better than non-attentional ones in handling long sentences.

Towards the end of this thesis, we explore the potential of NMT for GEC, as we believe that the distributed representation of words could help correct previously unseen errors more effectively than SMT. To the best of our knowledge, this is the first work to use the NMT framework to build end-to-end GEC systems.

2.3 Learner corpora

Unlike native corpora, learner corpora are collections of language data produced by non-native speakers. Having such learner resources is advantageous for GEC research: 1) it allows the investigation of real learner errors as well as the contexts in which they occur; 2) it facilitates the development of statistical models for GEC; for example, an SMT system requires millions of examples of correct and incorrect usage to learn reliable correction mappings; and 3) it provides a way of evaluating GEC system performance in a real world scenario. Recently, error-annotated learner corpora have become more readily and publicly available. In this section, we describe the learner corpora used in this thesis.

2.3.1 NUCLE

The National University of Singapore Corpus of Learner English (NUCLE) is an annotated corpus of learner text built by the National University of Singapore (NUS) NLP Group in collaboration with the NUS Centre for English Language Communication (Dahlmeier et al., 2013). It consists of more than 1,400 essays written by undergraduate students at NUS who are non-native English speakers. These essays were written in response to some prompts that cover a wide range of topics, such as environmental pollution, healthcare, and technology innovation. Two of the prompts used for data collection are shown below:

Prompt 1

> *"Public spending on the aged should be limited so that money can be diverted to other areas of the country's development." Do you agree?*

Prompt 2

> *Surveillance technology such as RFID (radio-frequency identification) should not be used to track people (e.g. human implants and RFID tags on people or products). Do you agree? Support your argument with concrete examples.*

Error type	**Prop.** (%)	**Example**
Wrong collocation/idiom/preposition (Wcip)	15.7	Singapore has invested heavily **on/in* the establishment of Biopolis.
Local redundancies (Rloc)	13.7	Abortion is available to end a life only **because of/because* the fetus or embryo has the wrong sex.
Article or determiner (ArtOrDet)	12.9	Sex selection technology should not be used in **non-medical/a non-medical* situation.
Noun number (Nn)	8.5	Sex selection should therefore be used for medical **reason/reasons*.
Mechanics (Mec)	7.1	The **affect/effect* of that policy has yet to be felt.
Verb tense (Vt)	7.1	A university **had conducted/conducted* the survey last year.
Word form (Wform)	4.8	Sex-selection may also result in **addition/additional* stress for the family.
Subject-verb agreement (SVA)	3.4	The boy **play/plays* soccer.
Verb form (Vform)	3.0	Will the child blame the parents after he **growing/grows* up?

Table 2.1: Proportion of the most common error types in the NUCLE corpus. Grammatical errors in the examples are printed in *italics* in the form **incorrect word/corrected word*.

The corpus contains over one million words which were manually annotated by professional English instructors at NUS using a tag set of 27 error categories (see Appendix A), resulting a total number of 46,597 error annotations. The statistics of NUCLE show that 57.6% of all sentences have no errors, 20.5% have exactly one error, 10.7% have exactly two errors, and 11.2% of all sentences have more than two errors. The highest observed number of error annotations in a single sentence is 28. The top nine error types in the NUCLE corpus are presented in Table 2.1. Although wrong word choice (*Wcip*) and redundancy errors (*Rloc*) are ranked at the top, Dahlmeier et al. (2013) reported that most *Wcip* errors are preposition errors, and a large percentage of *Rloc* errors involve articles that should be deleted. This confirms that articles and prepositions are two of the most common errors in ESL text. It also shows inconsistency in NUCLE labelling.[8]

2.3.2 CLC

The Cambridge Learner Corpus (CLC) is the world's largest learner corpus, developed by Cambridge University Press and Cambridge English Language Assessment since 1993. It is a 52.5 million word collection of exam scripts written by learners of English who took Cambridge English examinations around the world. Currently, it comprises over 200,000 exam scripts produced by learners at various levels speaking

[8]The NUCLE corpus was later revised for the CoNLL shared tasks to separate prepositions from *Wcip* and articles from *Rloc* (amongst other things) - see Section 2.5.2.

148 different L1s living in 217 different countries or territories. A subset of the corpus (a 25.5 million word collection) has been manually error coded by linguists using an error-coding system with a taxonomy of approximately 80 error types devised specifically for the CLC (Nicholls, 2003). The majority of the error codes used in the CLC are two-letter codes with the first letter representing the general type of error (e.g. spelling, word form) and the second one representing the word class (i.e. POS) of the required word (see Appendix B). The coding format is explained with the following examples:

Example 2.2. *I am so <NS type="RJ">exciting<c>excited</c></NS> that I have won the first prize.*

Example 2.3. *I like playing in <NS type="MD"><c>a</c></NS> team and deciding quickly what to do next.*

Error information is provided inside the *<NS>* tag, where the error type is also specified. Inside the *<NS>* tag, the original erroneous part is marked by the ** tag and its corrected version is marked by the *<c>* tag. In Example 2.2, *"RJ"* stands for Replace adJective, where 'exciting' should be corrected to 'excited'. In Example 2.3, *"MD"* stands for Missing Determiner, where the word 'a' should be added. Other error codes include Form, Unnecessary, Spelling and Derivation for the first letter; and Noun, Verb, preposiTion, Punctuation for the second letter. More detailed information about the error-coding scheme can be found in Nicholls (2003).

The top nine error types in the error-coded CLC are presented in Table 2.2, with spelling errors excluded. The most frequent error type in the CLC is choosing an inappropriate open class word (noun, verb, adjective or adverb), followed by prepositions and determiners.[9] A similar error distribution was observed in the NUCLE corpus - see Table 2.1.

Each examination script in the CLC contains meta-data about the learner, such as L1, nationality, age, sex and level of English, as well as the examination. There are three examination suites in the CLC (Williams, 2008):

- main suite (general purpose qualification):

  Certificate of Proficiency in English, Certificate of Advanced English, First Certificate in English (FCE), Preliminary English Test, and Key English Test;

- Business English Certificates (focuses on the language of business);

- International English Language Testing System (IELTS) (general and academic modules).

Two subsets of the CLC used in this thesis are described in detail: FCE and IELTS examination scripts.

[9]The determiner errors include both determiner and pre-determiner errors, not just the articles *a/an* and *the*.

Error type	**Prop.** (%)	**Example**
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	Wemust try our best to avoid **the/a* shortage of fresh water.
Comma error	9.3	However **/*, I'll 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.

Table 2.2: Proportion of the most common error types in the CLC. Grammatical errors in the examples are printed in *italics* in the form **incorrect word/corrected word*.

2.3.2.1 FCE examination scripts

The FCE dataset was released into the public domain in 2011 by Yannakoudakis et al. (2011). It is a set of 1,244 scripts written by learners of English who took the FCE examination between 2000 and 2001, which assesses English at an upper-intermediate level. The FCE dataset contains about half a million words and more than 50k errors. Each exam script contains two essays whose length varies between 120 and 180 words. Essays were written in response to tasks requiring a learner to write a letter, a report, an article, a composition or a short story. A typical prompt is shown below:

> *Your teacher has asked you to write a story for the school's English language magazine. The story must begin with the following words: "Unfortunately, Pat wasn't very good at keeping secrets".*

The anonymised scripts are annotated using XML and linked to meta-data including the question prompts and information about candidates.

2.3.2.2 IELTS examination scripts

The IELTS dataset is another subcorpus of the CLC that comprises exam scripts written by ESL learners taking the IELTS examination. It consists of 851 scripts from 2008 and 100 scripts from 2010. Like in the FCE dataset, each exam script in the IELTS dataset consists of two essays in response to two tasks. The first task asks a learner to write a descriptive report on the information provided in a diagram, table or short piece of text, or write a short letter in response to a situation or problem with a minimum of 150 words. The second task asks a learner to use at least 250 words to present an argument or discuss a problem.

2.4 Evaluation metrics

For system development, it is necessary to have internal system evaluation. Automatic evaluation metrics allow fast and inexpensive feedback. When evaluating a GEC system, the system's output is compared to gold-standard references provided by human experts. There is an on-going discussion on how to best evaluate GEC systems and several metrics have been proposed and used (Dale and Kilgarriff, 2011; Dale et al., 2012; Papineni et al., 2002; Dahlmeier and Ng, 2012b; Felice and Briscoe, 2015; Bryant and Ng, 2015; Napoles et al., 2015; Grundkiewicz et al., 2015; Sakaguchi et al., 2016). In this section, we present four evaluation metrics used in this thesis.

2.4.1 BLEU

BLEU was first proposed by Papineni et al. (2002) and is now used as the dominant method for automatic MT evaluation. It estimates the quality of the text produced by MT systems so that the closer it is to human translations, the better. BLEU has been shown to correlate well with human judgments at the corpus level. It uses a modified n-gram precision (p_{n}) to compare a candidate against multiple references:

$$p_n = \frac{\sum_{n\text{-}gram \in C} count_{clip}(n\text{-}gram)}{\sum_{n\text{-}gram \in C} count(n\text{-}gram)} \tag{2.1}$$

where C is a candidate sentence. The count of each n-gram in C is clipped by its maximum reference count observed in any single reference for that n-gram:

$$count_{clip} = min(count, max\_ref\_count) \tag{2.2}$$

BLEU is then defined as:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_{\mathrm{n}} \log p_{\mathrm{n}}\right) \tag{2.3}$$

where N is the order of the highest n-gram to be considered (usually $N = 4$); w_n stands for uniformly distributed weights: $w_{\mathrm{n}} = \frac{1}{N}$. BP is a *brevity penalty* which is used to prevent very short candidates from receiving very high scores:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \tag{2.4}$$

where c and r are the lengths of the system's candidate and gold-standard reference respectively.

BLEU was used by Mizumoto et al. (2011, 2012) to evaluate SMT-based GEC systems. Unlike metrics which rely on references with explicitly labelled error annotations, BLEU only requires corrected references. On the one hand, it can be used as a generic evaluation method independent of the annotation scheme, but on the other hand, it fails to provide detailed error type feedback for GEC. Since both the original and corrected sentences are in the same language (i.e. English) and most

words in the sentence do not need changing, BLEU scores for GEC systems are relatively high compared with standard MT tasks. However, it is not enough to just compare BLEU scores from different GEC systems, it is also necessary to compare them with that of the original input. If the system's ouput yields a higher BLEU score than the original input, it is assumed that the system improves the quality of the original input by making some corrections. In addition, BLEU allows multiple references, which is useful for errors with multiple alternative corrections.

2.4.2 M^2 scorer

The M^2 scorer, proposed by Dahlmeier and Ng (2012b), is used to evaluate system performance by how well its proposed corrections or edits match the gold-standard edits. It computes the sequence of phrase-level edits between a source sentence and a system's candidate that achieves the highest overlap with the gold-standard annotation. A parameter μ is used to limit the number of unchanged words (max_unchanged_words) so that edits including too many words are avoided. Evaluation is performed by computing precision (P), recall (R) and F-score (van Rijsbergen, 1979):

$$P = \frac{\sum_{i=1}^{n} |\mathbf{e}_\mathrm{i} \cap \mathbf{g}_\mathrm{i}|}{\sum_{i=1}^{n} |\mathbf{e}_\mathrm{i}|} \tag{2.5}$$

$$R = \frac{\sum_{i=1}^{n} |\mathbf{e}_\mathrm{i} \cap \mathbf{g}_\mathrm{i}|}{\sum_{i=1}^{n} |\mathbf{g}_\mathrm{i}|} \tag{2.6}$$

$$F_\beta = (1 + \beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R} \tag{2.7}$$

where $\mathbf{e_i} = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_\mathrm{n}\}$ is the system's candidate edit set and $\mathbf{g_i} = \{\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_\mathrm{n}\}$ is the gold-standard edit set. The intersection between $\mathbf{e_i}$ and $\mathbf{g_i}$ is defined as:

$$\mathbf{e}_\mathrm{i} \cap \mathbf{g}_\mathrm{i} = \{e \in \mathbf{e}_\mathrm{i} \mid \exists\ g \in \mathbf{g}_\mathrm{i}\, (match\,(e, g))\} \tag{2.8}$$

Two of the commonly used F-scores are F_1, which weights P and R evenly, and $F_{0.5}$, which emphasises P twice as much as R:

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{2.9}$$

$$F_{0.5} = (1 + 0.5^2) \times \frac{P \times R}{(0.5^2 \times P) + R} \tag{2.10}$$

The M^2 scorer was the official scorer in the CoNLL 2013 and 2014 shared tasks on grammatical error correction, where F_1 was used in CoNLL-2013 and $F_{0.5}$ was used in CoNLL-2014 (see Section 2.5.2). When building GEC systems, minimising the number of unnecessary corrections is often regarded as more important than covering a large number of errors, which is something users are willing to sacrifice as long as the system provides accurate corrections. In other words, high P is often preferred over high R. There is also a strong educational motivation, as flagging

Source	Candidate	Reference	Classification
a	a	a	TN
a	a	b	FN
a	a	-	FN
a	b	a	FP
a	b	b	TP
a	b	c	FP, FN, FPN
a	b	-	FP, FN, FPN
a	-	a	FP
a	-	b	FP, FN, FPN
a	-	-	TP
-	a	a	TP
-	a	b	FP, FN, FPN
-	a	-	FP
-	-	a	FN

Table 2.3: The extended writer-annotator-system evaluation scheme proposed by Felice and Briscoe (2015).

correct text as incorrect would cause confusion among learners. This is why $F_{0.5}$ was much preferred lately when reporting system performance.

However, evaluation methods based on P, R and F-score (e.g. the M^2 scorer) do not provide an indicator of improvement on the original text so there is no way to compare GEC systems against a 'do-nothing' baseline that keeps the input text unchanged. A 'do-nothing' baseline will always yield an F-score of 0 by definition, and an increase in P, R or F-score does not necessarily mean a reduction in the actual error rate.

2.4.3 I-measure

The I-measure was designed by Felice and Briscoe (2015) to address problems with previous evaluation methods and to evaluate real improvement on the original sentence after corrections.

System performance is first evaluated in terms of weighted accuracy (WAcc), based on a token-level alignment between a source sentence, a system's candidate, and a gold-standard reference. An extended version of the writer-annotator-system evaluation scheme (Chodorow et al., 2012) was adopted where each token alignment is classified as a true positive (TP), true negative (TN), false positive (FP), false negative (FN), or both an FP and FN (FPN) - see Table 2.3. WAcc is defined as:

$$WAcc = \frac{w \cdot \mathrm{TP} + \mathrm{TN}}{w \cdot (\mathrm{TP} + \mathrm{FP}) + \mathrm{TN} + \mathrm{FN} - (w+1) \cdot \frac{\mathrm{FPN}}{2}} \tag{2.11}$$

where $w > 1$ is a weight factor.

An *Improvement* or I-measure (I) score is computed by comparing system performance (WAcc_{sys}) with that of a baseline that leaves the original text unchanged ($\text{WAcc}_{\text{base}}$):

$$I = \begin{cases} \lfloor WAcc_{\text{sys}} \rfloor & \text{if } WAcc_{\text{sys}} = WAcc_{\text{base}} \\ \dfrac{WAcc_{\text{sys}} - WAcc_{\text{base}}}{1 - WAcc_{\text{base}}} & \text{if } WAcc_{\text{sys}} > WAcc_{\text{base}} \\ \dfrac{WAcc_{\text{sys}}}{WAcc_{\text{base}}} - 1 & \text{otherwise} \end{cases} \tag{2.12}$$

Values of I lie in the [-1, 1] interval.[10] Positive values indicate improvement, while negative values indicate degradation. A score of 0 indicates no improvement (i.e. baseline performance), 1 indicates 100% correct text and -1 indicates 100% incorrect text.

As multiple annotations are taken into account, the I-measure is computed after maximising WAcc_{sys} at the sentence level, so as to ensure all the evaluated hypotheses are paired with their highest scoring references. Trying to maximise I score directly can yield suboptimal results, as different combinations of $\text{WAcc}_{\text{base}}$ and WAcc_{sys} can produce the same final result (but the one with higher WAcc_{sys} is clearly preferred).

2.4.4 GLEU

Generalized Language Evaluation Understanding (GLEU), proposed by Napoles et al. (2015), is a simple variant of BLEU for GEC which takes the original source into account. GLEU modifies the n-gram precision in BLEU to assign extra weight to n-grams present in the candidate that overlap with the reference but not the source ($R \backslash S$), and penalise those in the candidate that are in the source but not the reference ($S \backslash R$). For a correction candidate C with a corresponding source S and reference R, the modified n-gram precision ($p_{\text{n}}{}'$) for $GLEU(C, R, S)$ is defined as:

$$p_{\text{n}}{}' = \frac{\sum_{n\text{-}gram \in C} count_{R \backslash S}(n\text{-}gram) - \lambda(count_{S \backslash R}(n\text{-}gram)) + count_R(n\text{-}gram)}{\sum_{n\text{-}gram \in C} count_S(n\text{-}gram) + \sum_{n\text{-}gram \in R \backslash S} count_{R \backslash S}(n\text{-}gram)} \tag{2.13}$$

where the weight λ determines by how much incorrectly changed n-grams are penalised. Given a bag of n-grams B, the counts in Equation 2.13 are collected as:

$$count_B(n\text{-}gram) = \sum_{n\text{-}gram' \in B} d(n\text{-}gram, n\text{-}gram') \tag{2.14}$$

$$d(n\text{-}gram, n\text{-}gram') = \begin{cases} 1 & \text{if } n\text{-}gram = n\text{-}gram' \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

[10] I score is often expressed as a percentage.

By updating the n-gram precision in Equation 2.3, GLEU is defined as:[11]

$$GLEU(C, R, S) = BP \cdot \exp\left(\sum_{n=1}^{N} w_{\mathrm{n}} \log {p_{\mathrm{n}}}'\right) \tag{2.16}$$

Similar to BLEU, GLEU works with multiple references at the corpus level. It can be used as a generic evaluation method independent of the annotation scheme, but fails to provide detailed system performance for each error type.

2.5 Shared tasks on grammatical error correction

In the last few years, four GEC shared tasks have provided a forum for participating teams to compare results on common training and test data. Participants are provided with a fully annotated training set and encouraged to use any publicly available data and tools to build their GEC systems in a few months' time. After that, new blind test data is used to evaluate system performance for the participating teams. Systems are expected to detect grammatical errors in text written by non-native speakers and return corrected versions within a few days after the release of the test data. The organisers then evaluate each system's output and release the final rankings.

2.5.1 HOO 2011 and 2012

The first two shared tasks - Helping Our Own (HOO) 2011 and 2012 - were aimed to promote the use of NLP tools and techniques for the development of automated systems that could provide writing assistance to non-native authors in the NLP community (Dale and Kilgarriff, 2011; Dale et al., 2012). In the HOO-2011 shared task, participants were provided with a set of documents extracted from the ACL Anthology[12] written by non-native authors. The task was to automatically detect and correct all errors present in text. Errors were classified into 13 error types based on the CLC coding system (Nicholls, 2003). Six teams participated in the task, with some achieving top performance by focussing only on a limited number of error types.

Given the difficulty of HOO-2011, the HOO-2012 shared task focussed only on article and preposition errors. The FCE dataset was provided as the official training set. The number of participating teams increased to 14 and most participants built machine learning classifiers. Evaluation in both HOO shared tasks was performed by computing P, R and F-score between a system's edit set and a manually created gold-standard edit set.

[11] We notice that there is a new version of GLEU appeared this year (Napoles et al., 2016b). However, scores reported in this thesis were computed using the original GLEU (Napoles et al., 2015) described in this section.

[12] A digital archive of research papers in computational linguistics: `https://aclweb.org/anthology`

2.5.2 CoNLL 2013 and 2014

The next two shared tasks took place in conjunction with the Conference on Computational Natural Language Learning (CoNLL). The CoNLL-2013 shared task (Ng et al., 2013) expanded the scope of HOO-2012 to include three new error types: noun number (*Nn*), verb form (*Vform*) and subject-verb agreement (*SVA*). Together with article (*ArtOrDet*) and preposition (*Prep*) errors, this new error list is more comprehensive and also introduces interacting errors. NUCLE v2.3[13] was used as in-domain training data. The test data consists of 50 new essays, which were written in response to two prompts. One prompt was also used in the training set, while the other was new.

Systems were evaluated using the M^2 scorer with the max_unchanged_words parameter μ set to 3 as suggested by the organisers (limiting the maximum unchanged words to three per edit). Rankings were based on F_1, weighting P and R equally. Initially, there was only one set of gold annotations, but since there are often multiple valid corrections for some errors, participating teams were subsequently allowed to propose alternative answers (gold-standard edits). This practice was adopted from the HOO 2011 and 2012 shared tasks. Therefore, there were two rounds of evaluation, the second of which allowed alternative answers. As noted by Ng et al. (2013), these new scores tended to be biased towards the teams which submitted alternative answers. Consequently, to reduce bias, they suggested future evaluation be carried out without alternative answers. In the end, 17 teams participated in CoNLL-2013. Among these teams, a common approach was to build classifiers for each error type. Other approaches included LM, MT and heuristic rules.

The CoNLL-2014 shared task (Ng et al., 2014) tried to once again push the boundaries of GEC by returning to an all-errors correction task. In particular, there were three major changes compared with CoNLL-2013: 1) participating systems were expected to correct grammatical errors of all types; 2) two human annotators annotated the test essays independently; and 3) the evaluation metric was changed from F_1 to $F_{0.5}$, to prioritise P over R. A newer version of the NUCLE corpus - NUCLE v3.0 - was used as official training data. Additionally, a new set of 50 essays written by non-native English speakers was used as blind test data. The CoNLL-2013 test set could be freely used for training and/or development. The M^2 scorer was again used as the official scorer.

In total, 13 teams submitted output to CoNLL-2014. Most of them built hybrid systems that combined different approaches. For non-specific error type correction, LM and MT approaches were used; whereas for single error types, rule-based approaches and machine learning classifiers were preferred. We built a phrase-based SMT system (see Chapter 3) which contributed to our final hybrid system submitted to the shared task. Our system achieved the best $F_{0.5}$ and R on the original evaluation.

[13]In NUCLE v2.3, 17 essays were removed from the first release of NUCLE, and the error types *Wcip* and *Rloc* were mapped to *Prep, Wci, ArtOrDet*, and *Rloc-* using POS tags.

2.6 Use of datasets and evaluation metrics in this thesis

In the rest of this thesis, different learner corpora and evaluation metrics are used in different chapters. This is because, according to Chodorow et al. (2012), there is no single best evaluation metric and the usefulness of a metric depends on the application and research goals.

The work in Chapter 3 is presented in the context of the CoNLL-2014 shared task. The NUCLE corpus, as provided by the shared task organisers, is used as in-domain training data, and results are reported on the CoNLL-2014 development set (i.e. the CoNLL-2013 test set) and test set. $F_{0.5}$ as calculated by the M^2 scorer is used as the evaluation measure. Parallel sentences extracted from the publicly available FCE dataset and the IELTS dataset are used as additional training data.

In Chapter 4 and 5, the publicly available FCE dataset is used and results are reported on the FCE test set. The reasons for using the FCE dataset rather than NUCLE are manifold. Firstly, the FCE dataset, as a subcorpus of the CLC, covers a wide variety of L1s and was used in the HOO-2012 error correction shared task. Compared with the NUCLE corpus used in the CoNLL 2013 and 2014 shared tasks, which only contains essays written by undergraduate students at NUS, the FCE dataset is a more representative test set of learner writing. Secondly, the error annotations in the NUCLE corpus are sometimes unreliable and inconsistent. This may introduce noise into final GEC systems and result in underestimations of performance. Thirdly, as described in Section 2.3.2, the CLC is the world's largest learner corpus, and the FCE dataset was annotated using the same annotation scheme as the CLC. In order to make better use of the CLC and avoid any problems caused by the inconsistency between different annotation schemes, we use the FCE dataset in our experiments. Results reported on the publicly available FCE dataset can be used for cross-system comparisons.

As discussed in Section 2.4, evaluation methods based on P, R and F-score (e.g. the M^2 scorer) do not provide an indicator of improvement on the original text. Given this, and the fact that an increase in P, R or F-score does not necessarily mean a reduction in the actual error rate, we opt to use the I-measure and thus gain a better insight into system performance in terms of improvement on the original text.

We also apply our CLC-trained systems to the CoNLL-2014 shared task development and test sets directly, without using or optimising for NUCLE. Results are reported using BLEU, GLEU, $F_{0.5}$ (calculated by the M^2 scorer) and I-measure for broader cross-system comparisons.

CHAPTER 3

Building a preliminary SMT-based GEC system

In this chapter, we investigate whether SMT can form the basis of a competitive all-errors GEC system and describe how to build an end-to-end system using the SMT framework. Our SMT method is evaluated in the context of the CoNLL-2014 all-errors correction shared task (see Section 2.5.2) and the results presented in Section 3.5 were previously published in Felice et al. (2014). Additionally, the work on artificial data described in Section 3.3.4.2 was further developed in Felice and Yuan (2014b).

3.1 Statistical machine translation

As one of the first applications envisaged for computers, MT is the process of translating a sentence from one language into another automatically. Several approaches have been proposed, such as word-for-word translation, interlingual approaches (Mitamura et al., 1991), example-based MT (Nagao, 1984; Sato and Nagao, 1990) and SMT. In the last two decades, SMT has become the dominant approach both in the research community and in the commercial sector. The underlying idea is that language is so complex and productive that it could never be fully analysed and distilled into a set of rules. Instead, a machine should try to learn translation mappings automatically from large parallel corpora by pairing the input and output of the translation process and learning from the statistics over the data, thus removing the need for linguists or language experts. SMT stands out for its low cost and rapid prototyping, which also produces state-of-the-art results for many MT tasks.

GEC can be considered a special case of MT where the task is to translate 'bad' English sentences into 'good' ones. The SMT approach to GEC has several advantages. First, an SMT system can deal with multiple error types at the same time without having to (a) classify errors into different types, (b) decide on their boundaries, or (c) combine the results of multiple classifiers, which is often the case in traditional machine learning approaches. Interacting errors are expected to be corrected as well since SMT systems work at a global rather than local level (compared

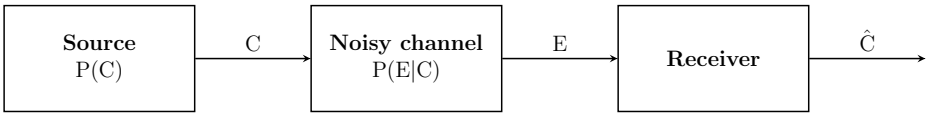


Figure 3.1: A noisy channel model.

with classifiers) and corrections are performed jointly for the entire sentence. As a result, the SMT approach seems more suitable for an all-errors task. Second, it does not need expert knowledge but only requires a parallel corpus of grammatically incorrect sentences as the source and their corrected versions as the target. Following the principle that 'more data is better data' (Church and Mercer, 1993), previous studies in SMT have shown that the larger the training parallel corpus, the more reliable the translation mappings that can be learnt and the better the translation performance that can be achieved (Koehn et al., 2003; Suresh, 2010; Axelrod et al., 2011). Last but not least, there are well-developed tools for SMT which can be easily adapted for GEC, so we can benefit from state-of-the-art SMT techniques.

An SMT-based GEC system can be modelled as a machine translator. Here, the input (source) is an erroneous English sentence $E = e_1\ e_2\ ...\ e_m$, and the output (target) is a corrected sentence $C = c_1\ c_2\ ...\ c_l$. The erroneous sentence E produced by ESL learners can be regarded as having passed through a noisy channel (Shannon, 1948) and is hence corrupted by noise, i.e. interference from learners' L1s - see Figure 3.1. The goal is to recover the sentence C based on the corrupt sentence E:

$$\hat{C} = \arg\max_C P(C|E) = \arg\max_C \frac{P(E|C)P(C)}{P(E)} = \arg\max_C P(E|C)P(C) \qquad (3.1)$$

where $P(E)$ in the denominator is ignored since it is a constant across all Cs. The other three parameters that need to be considered are:

- LM:

  estimates the corrected sentence probability $P(C)$ from a target language corpus;

- TM:

  estimates the translation (i.e. correction) probability $P(E|C)$ from a parallel corpus;

- decoder:

  searches for the target sentence C that maximises the product of $P(C)$ and $P(E|C)$.

3.1.1 The language model

One essential component of any SMT system is the LM, which makes sure that the final system outputs fluent English sentences. A LM is a function that takes an English sentence as input and returns the probability that it is a valid English sentence.

3.1.1.1 N-gram language model

Statistical LMs were designed to assign a probability to a sequence of words (or tokens, which may include punctuations, etc.) based on counts from a training corpus (Brants et al., 2007). Since most long sequences of words will never occur in the text at all, the process of predicting a word sequence is broken down into predicting one word at a time. A sentence C with l words can be decomposed using the chain rule (Koehn, 2010):

$$P(C) = P(c_1, c_2, ..., c_l) = \prod_{i=1}^{l} P(c_i|c_1, c_2, ..., c_{i-1}) \tag{3.2}$$

Following the Markov assumption, Equation 3.2 can be reformulated to only consider the most recent $(n-1)$ words when predicting the next word:

$$P(C) = P(c_1, c_2, ..., c_l) \approx \prod_{i=1}^{l} P(c_i|c_{i-n+1}, ..., c_{i-1}) \tag{3.3}$$

For a unigram LM ($n = 1$), the probability depends only on the current word:

$$P(C) = P(c_1, c_2, ..., c_l) \approx \prod_{i=1}^{l} P(c_i) \tag{3.4}$$

where

$$P(c_i) = \frac{count(c_i)}{N} \tag{3.5}$$

$count(c_i)$ is the number of times c_i is seen in the training corpus and N is the total number of words seen in the same corpus.

For a bigram LM ($n = 2$), the probability is calculated as:

$$P(C) = P(c_1, c_2, ..., c_l) \approx \prod_{i=1}^{l} P(c_i|c_{i-1}) \tag{3.6}$$

where

$$P(c_i|c_{i-1}) = \frac{count(c_{i-1}^{i})}{count(c_{i-1})} \tag{3.7}$$

For a higher-order n-gram LM:

$$P(C) = P(c_1, c_2, ..., c_l) \approx \prod_{i=1}^{l} P(c_i|c_{i-n+1}^{i-1}) \tag{3.8}$$

where

$$P(c_i|c_{i-n+1}^{i-1}) = \frac{count(c_{i-n+1}^{i})}{count(c_{i-n+1}^{i-1})} = \frac{count(c_{i-n+1}^{i})}{\sum_{c_i} count(c_{i-n+1}^{i})} \tag{3.9}$$

Higher-order n-grams can capture information about longer sequences, while the choice of n typically depends on the size of the training corpus.

Ideally, correct word sequences will get high probabilities while incorrect or unseen sequences will get low probabilities. However, no matter how large a training corpus is, it is impossible to cover all possible correct word sequences. As we move towards higher-order n-grams, data sparsity becomes a more serious problem and we are more likely to encounter unseen n-grams. Equation 3.9 assigns unseen n-grams a probability of 0 and is undefined when the denominator is 0. N-grams with zero counts will then result in a zero probability for the whole string. Since we do not want to give any string zero probability, we need a way to assign some probabilities to unseen n-grams. In practice, *smoothing* and *back-off* techniques are often used.

Smoothing methods are used to assign positive probabilities to unseen n-grams, e.g. *add-one smoothing* (Laplace, 1825; Lidstone, 1920; Johnson, 1932; Jeffreys, 1961) and *Good-Turing smoothing* (Good, 1953). However, they assign all unseen n-grams the same probability, making no distinction between them.

Another option is to *back off* to lower order n-grams with richer and more reliable statistics. When estimating the probability of an n-gram c_{i-n+1}^{i}, if we have seen the n-gram in the training corpus (i.e. $count(c_{i-n+1}^{i}) > 0$), we use the raw LM probability $P(c_i|c_{i-n+1}^{i-1})$; otherwise we *back off* to the lower order probability $P_{bo}(c_i|c_{i-(n-1)+1}^{i-1})$:

$$P_{bo}(c_i|c_{i-n+1}^{i-1}) = \begin{cases} d(c_{i-n+1}^{i-1})P(c_i|c_{i-n+1}^{i-1}) & \text{if } count(c_{i-n+1}^{i}) > 0 \\ \alpha(c_{i-n+1}^{i-1})P_{bo}(c_i|c_{i-(n-1)+1}^{i-1}) & \text{otherwise} \end{cases} \tag{3.10}$$

A discounting function d ($0 \leq d \leq 1$) is introduced to ensure that overall probabilities add up to 1 for a history c_{i-n+1}^{i-1}. One way to compute d is to first group histories based on their counts in the corpus. If we have seen the history very frequently, we would trust predictions based on this history more, and therefore set a fairly high value for d. Otherwise, we give more weight to the *back-off* probability through α, resulting in a small d.

3.1.1.2 Kneser-Ney smoothing

Kneser-Ney smoothing introduced new ways of constructing the higher-order and lower-order models used in the *back-off* model. For the higher-order model, *absolute discounting* (Ney and Essen, 1991; Ney et al., 1994) is used to reduce the probability mass for observed n-grams. Rather than using the discounting function d in Equation 3.10, a fixed discount D ($0 \leq D \leq 1$) is subtracted from non-zero counts:

$$P(c_i|c_{i-n+1}^{i-1}) = \frac{max\left\{count(c_{i-n+1}^{i}) - D, 0\right\}}{\sum_{c_i} count(c_{i-n+1}^{i})} \tag{3.11}$$

Ney et al. (1994) proposed a way to calculate D:

$$D = \frac{N_1}{N_1 + N_2} \tag{3.12}$$

where N_1 and N_2 are the total numbers of n-grams with exactly one and two counts respectively in the training data.

We notice that the lower-order model is useful only when counts in the higher-order model are small or zero. Therefore, the lower-order model should be optimised to perform well in these situations. However, the *back-off* lower-order model defined in Equation 3.10 does not distinguish words that are very frequent but only occur in a restricted set of contexts from those which are less frequent but occur in many more contexts. In *Kneser-Ney smoothing*, the lower-order model is modified to take the diversity of histories into account, where the raw count is replaced with the count of histories for a word:

$$P_{KN}(c_i|c_{i-(n-1)+1}^{i-1}) = \frac{N_{1+}(\cdot c_{i-(n-1)+1}^{i})}{N_{1+}(\cdot c_{i-(n-1)+1}^{i-1}\cdot)} \tag{3.13}$$

where

$$N_{1+}(\cdot c_{i-(n-1)+1}^{i}) = |\{c_{i-n+1} : count(c_{i-n+1}^{i}) > 0\}| \tag{3.14}$$

$$N_{1+}(\cdot c_{i-(n-1)+1}^{i-1}\cdot) = \sum_{c_i} N_{1+}(\cdot c_{i-(n-1)+1}^{i}) \tag{3.15}$$

The *back-off* model is then defined as:

$$P_{KN}(c_i|c_{i-n+1}^{i-1}) = \begin{cases} \frac{max\{count(c_{i-n+1}^{i}) - D, 0\}}{\sum_{c_i} count(c_{i-n+1}^{i})} & \text{if } count(c_{i-n+1}^{i}) > 0 \\ \gamma(c_{i-n+1}^{i-1}) P_{KN}(c_i|c_{i-(n-1)+1}^{i-1}) & \text{otherwise} \end{cases} \tag{3.16}$$

where $\gamma(c_{i-n+1}^{i-1})$ is chosen to make the probabilities sum to 1.

3.1.1.3 Modified Kneser-Ney smoothing

Modified Kneser-Ney smoothing is perhaps the best *smoothing* method widely used today. Chen and Goodman (1998) made three main changes to *Kneser-Ney smoothing*: 1) *interpolation* is used instead of *back-off*; 2) rather than using a single discount D for all non-zero counts, three different discount parameters, D_1, D_2, and D_{3+}, are used for n-grams with one, two, and three or more counts respectively; and 3) estimation for discount Ds is performed on held-out data. Equation 3.16 is modified to:

$$P_{MKN}(c_i|c_{i-n+1}^{i-1}) = \frac{count(c_{i-n+1}^{i}) - D(count(c_{i-n+1}^{i}))}{\sum_{c_i} count(c_{i-n+1}^{i})} + \gamma(c_{i-n+1}^{i-1}) P_{MKN}(c_i|c_{i-(n-1)+1}^{i-1}) \tag{3.17}$$

where

$$D(count) = \begin{cases} 0 & \text{if } count = 0 \\ D_1 & \text{if } count = 1 \\ D_2 & \text{if } count = 2 \\ D_{3+} & \text{if } count \geq 3 \end{cases}$$

To make the probabilities sum to 1, we take

$$\gamma(c_{i-n+1}^{i-1}) = \frac{D_1 N_1(c_{i-n+1}^{i-1}\cdot) + D_2 N_2(c_{i-n+1}^{i-1}\cdot) + D_{3+} N_{3+}(c_{i-n+1}^{i-1}\cdot)}{\sum_{c_i} count(c_{i-n+1}^{i})} \tag{3.18}$$

where

$$N_r(c_{i-n+1}^{i-1}\cdot) = |\{c_i : count(c_{i-n+1}^{i}) = r\}| \tag{3.19}$$

Optimal discount parameters D can be computed as:

$$\begin{aligned} D_1 &= 1 - 2Y\frac{N_2}{N_1} \\ D_2 &= 2 - 3Y\frac{N_3}{N_2} \\ D_{3+} &= 3 - 4Y\frac{N_4}{N_3} \end{aligned} \tag{3.20}$$

where

$$Y = \frac{N_1}{N_1 + 2N_2} \tag{3.21}$$

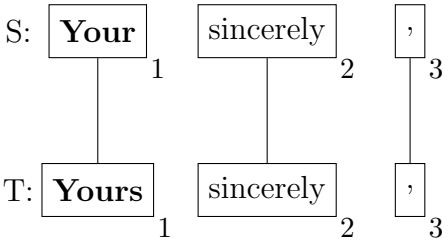
All these modifications have been proved to improve performance, making *modified Kneser-Ney smoothing* the best LM estimator.

3.1.2 The translation model

TMs are learnt from parallel corpora. However, unlike in LMs, it is not feasible to use simple word counts so a word alignment model must be introduced. An alignment can be formalised with an alignment function a. This function maps each output word at position i to an input word at position j:

$$a : j \rightarrow i \tag{3.22}$$

Example 3.1. In this source-target pair of sentences:



the alignment function a provides the following mappings:

$$a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3\} \tag{3.23}$$

The TM probability is then defined using the alignment model as:

$$P(E|C) = \sum_{a} P(a, E|C) \tag{3.24}$$

A sentence is broken up into chunks:

$$P(a, E|C) = \prod_{j=1}^{m} t(e_j|c_i) \tag{3.25}$$

where c_i is the chunk in the corrected sentence corresponding to the chunk e_j in the erroneous sentence and $t(e_j|c_i)$ is the probability of these two chunks being aligned. Relative frequency estimates can then be used to estimate the probability $t(e_j|c_i)$.

3.1.2.1 IBM Models 1-5

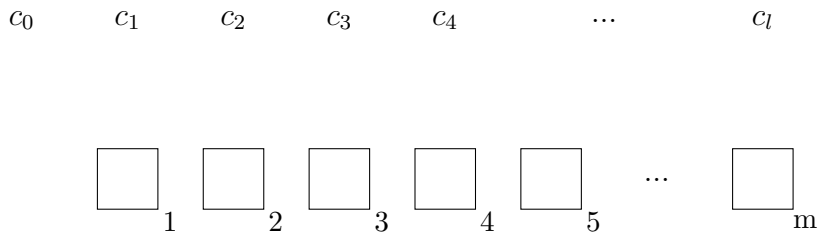
Initially, each chunk is made up of just one word, so the model obtained is based on word-to-word translation. Since we do not have word-aligned data (only sentence-aligned data is available in a parallel corpus), the Expectation-Maximisation (EM) algorithm (Dempster et al., 1977) is used to find the maximum likelihood estimation at the word level. The EM algorithm works as follows:

1. initialise the model, typically with uniform distributions;
2. apply the model to the data (expectation step);
3. learn the model from the data (maximisation step);
4. iterate steps 2 and 3 until convergence.

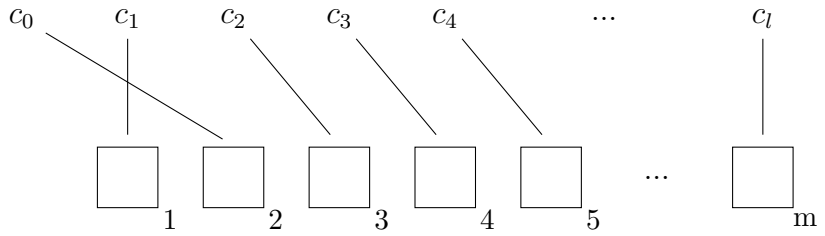
IBM Models 1-5 (Brown et al., 1993) and a Hidden Markov Model (HMM) (Vogel et al., 1996) define different decompositions of the probability $P(E|C)$ with different alignments a.

In IBM Models 1 and 2:

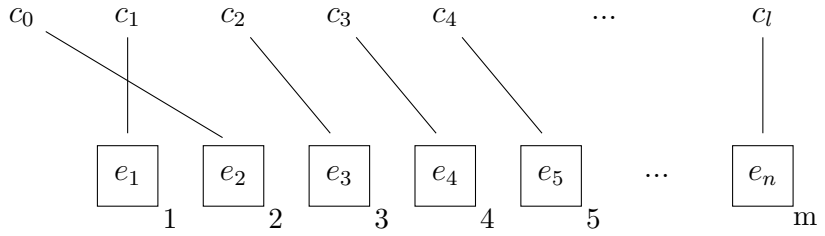
1. the length of the erroneous string m is first chosen (assuming all lengths have equal probability);



2. each position in the erroneous string is then connected to the word in the corrected string;



3. the erroneous word for each position in the erroneous string is finally decided.

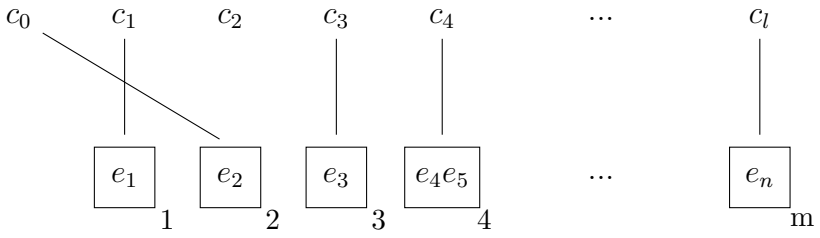


IBM Model 1 only uses lexical translation and assumes all connections to be equally likely, therefore the order of the words in C and E has no impact. The EM algorithm is used to estimate lexical translation probabilities. IBM Model 2 makes more realistic assumptions and adds an absolute reordering model. It addresses the issue of alignment with an explicit alignment model which depends on the positions it connects ($j \rightarrow i$) and the lengths of the input and output strings, that are m and l respectively:

$$a(i|j, m, l) \tag{3.26}$$

The HMM assumes that words do not move independently of each other and that the probability of a connection also depends on the previous connection position.

In Models 3, 4 and 5, a model of fertility (one-to-many mapping) is introduced so that one source word can be translated into zero or more target words when computing $P(E|C)$. Thus, a single word in the corrected string C can be aligned with zero, one or more words in the erroneous string E:



where $c_2 \rightarrow \varnothing$ and $c_4 \rightarrow e_4e_5$.

A relative reordering model is used in Model 4, where the probability of a connection also depends on previously translated words. Model 5 fixes deficiencies in Models 3 and 4 by keeping track of available positions, therefore avoiding probability mass on non-strings (i.e. impossible translations).

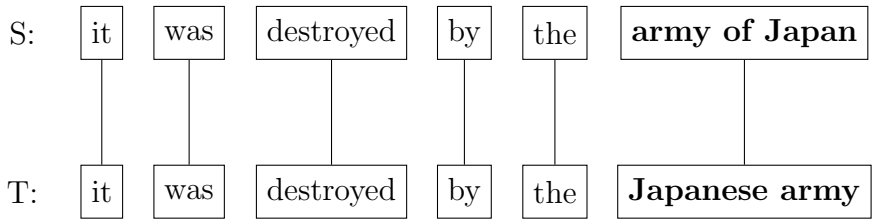
IBM Model 1 can be considered a special case of Model 2. Models 1-4 serve as stepping stones to Model 5. In addition, only Model 1 has a global maximum, which can be used as an initial estimate for the improved models.

3.1.2.2 Phrase-based models

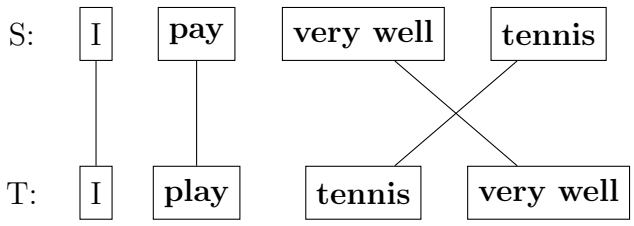
Word-based models like IBM Models 1-5 and the HMM use words as translation units and only allow one-to-one and one-to-many mappings, so they fail to represent many-to-one or many-to-many mappings that are common in translation tasks. Phrase-based models that allow many-to-many mappings are therefore widely used in current SMT applications (Koehn et al., 2003). Unlike in word-based models, phrases are used as translation units, so local contexts can be learnt and used during translation.

In phrase-based models, a source sentence E is first segmented into m phrases $\bar{e}_1^m$, where phrases can be any sequences of words. Each source phrase $\bar{e}_j$ is then translated into a target phrase $\bar{c}_i$ and target phrases can be further reordered, as in the following examples:

Example 3.2. Many-to-many mappings:



Example 3.3. Reodering:



The segmentation process is not modelled explicitly, and each segmentation is first assumed to be equally likely. A phrase translation table containing phrase mappings and their translation probabilities is then built and finally used during translation.

There are several ways to construct a phrase translation table and one of them is to build a table from a word alignment as follows:

1. create a word alignment for each sentence pair in the parallel corpus using IBM Models and the HMM;

2. extract phrase pairs from the word alignment using heuristic rules;

3. estimate phrase translation probabilities using relative frequency.

When extracting phrase pairs, both short and long phrases are collected. Shorter phrases are more likely to be used to translate previously unseen sentences while longer phrases capture more contextual information and can be used to translate large chunks of text at once. However, extracting phrases of any length results in a huge number of phrase pairs and a large phrase translation table. Even for well-behaved alignments without reordering, the number of extracted phrases is roughly quadratic with respect to the number of words (Koehn et al., 2003). Since most long phrases observed in the training data never occur in the test data, a maximum phrase length is set to reduce the number of extracted phrases and therefore keep the final phrase translation table manageable.

To compute conditional probability distributions for the phrase table, a number of phrase pairs for each sentence pair are first extracted. Then, the number of sentence pairs that include a particular phrase pair $(\bar{e}, \bar{c})$ is counted ($count(\bar{e}, \bar{c})$). Finally, the phrase translation probability $\phi(\bar{e}|\bar{c})$ is estimated using relative frequency:

$$\phi(\bar{e}|\bar{c}) = \frac{count(\bar{e}, \bar{c})}{\sum_{\bar{e}_j} count(\bar{e}_j, \bar{c})} \tag{3.27}$$

3.1.3 The reordering model

In phrase-based translation, a reordering or distortion model is used to handle phrase reorderings. As some phrases are reordered more frequently than others, we may want to learn a reordering preference for each phrase pair. A lexicalised reordering model that conditions reordering on the actual phrases can be learnt during phrase extraction. Three types of orientation (reordering) are defined:

- monotone (m):

  if a word alignment point to the top left exists;

- swap (s):

  if a word alignment point to the top right exists;

- discontinuous (d):

  if no word alignment point exists to the top left or the top right.

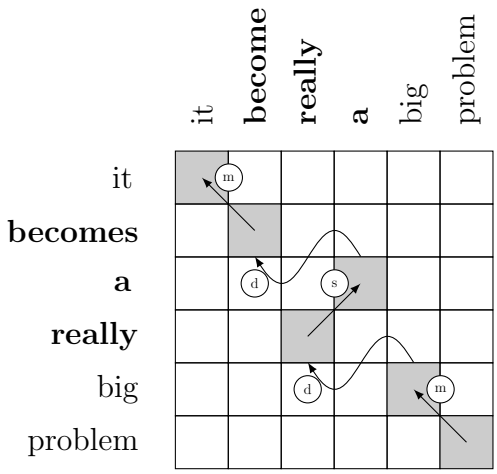


Figure 3.2: Three types of orientation in the lexicalised reordering model.

An example is given in Figure 3.2 to illustrate these three types of orientation. We count how many times each extracted phrase pair $(\bar{e}, \bar{c})$ is found with each of the three orientation types. A probability distribution P_o that predicts an orientation type (o) is estimated based on these counts:

$$P_o(o|\bar{e}, \bar{c}) = \frac{count(o, \bar{e}, \bar{c})}{\sum_o count(o, \bar{e}, \bar{c})} \tag{3.28}$$

where $o \in \{m, s, d\}$.

Due to sparse statistics of these orientation types, we *smooth* the counts in Equation 3.28 with a factor σ, so that:

$$P_o(o|\bar{e}, \bar{c}) = \frac{\sigma P_o(o) + count(o, \bar{e}, \bar{c})}{\sigma + \sum_o count(o, \bar{e}, \bar{c})} \tag{3.29}$$

where

$$P_o(o) = \frac{\sum_{\bar{e}} \sum_{\bar{c}} count(o, \bar{e}, \bar{c})}{\sum_o \sum_{\bar{e}} \sum_{\bar{c}} count(o, \bar{e}, \bar{c})} \tag{3.30}$$

3.1.4 The decoder

Decoding is the process of choosing the best translation from a pool of all possible candidate translations. The decoding problem in SMT is solved using *beam search*. During decoding, partial translations (i.e. hypotheses) are constructed. The process starts with an empty hypothesis that has no translations (i.e. corrections) of the input words. It then generates new hypotheses by extending the current hypothesis with phrase translations for input words that have not been translated yet. Different untranslated input words can be picked and translated, resulting in many different hypotheses. This hypothesis expansion is carried out recursively until all the words in the erroneous input sentence have been covered. Scores are computed incrementally for these hypotheses using the three models discussed in previous sec-

tions. Among these complete hypotheses (a.k.a. candidates), the highest scoring one is selected as the final translation output.

As we might expect for phrase-based translation, the fact that there are multiple ways to segment the source sentence into phrases and that each phrase can be translated differently will make the system produce a large number of hypotheses. The size of the search space grows exponentially with the length of the input sentence. This makes it computationally prohibitive to translate any long sentences. Likewise, it is too expensive to exhaustively examine all candidates to select the best. This complexity problem is addressed by *hypothesis recombination* and *pruning* (Koehn, 2010).

Hypothesis recombination takes advantage of the fact that matching hypotheses can be generated through different paths. Hypotheses with the same output words that cover the same erroneous input words are matching hypotheses. If multiple paths lead to matching hypotheses, any worse ones (i.e. with lower scores) are dropped. This process is *risk-free* as it only drops identical partial translations while keeping all the different hypotheses. *Hypothesis recombination* reduces the search space and avoids considering multiple paths that differ only in internal representations (i.e. different phrase segmentations).

Pruning is more efficient as it removes bad hypotheses at an early stage. Hypotheses are put into stacks according to some criteria (e.g. the number of erroneous input words being translated) and bad ones are discarded when these stacks get too big. *Beam search* is applied and a fixed threshold α is introduced: if a hypothesis is α times worse than the best hypothesis in the stack, it is pruned out. This is a *risky* expansion, however, since hypotheses that are considered 'bad' at an old stage may outperform those which are considered 'good' at a new stage. Therefore, a future cost of untranslated parts should be taken into account (i.e. how expensive it is to translate the rest of the input sentence).

3.2 Challenges in applying SMT to GEC

Statistical models like SMT benefit from a large amount of high-quality training data. For some well-defined MT tasks, many resources are available, e.g. parallel corpora or online bilingual data. In order to build good GEC systems using the SMT framework, we need a substantial amount of parallel training examples containing original erroneous sentences written by non-native English writers and their corrected versions, to make sure systems learn reliable corrections. Creating this kind of annotated learner corpora is a slow and costly process, as we often need linguists to manually correct all the errors in non-native text. In addition, the quality of the training data matters. Annotation errors may introduce noise into the final system and differences between training and test sets may result in low performance. The first difficulty lies in how to obtain sufficient high-quality training data at a reasonable cost. In this chapter, we investigate the use of three monolingual datasets for building LMs (Section 3.3.3) and three different sources of parallel data for building TMs (Section 3.3.4) using SMT.

Since phrase-based MT uses only lexical representations, newer advanced TMs have been proposed to overcome this limitation. Factored models allow the addition of extra linguistic information by representing each word with its lemma or POS tag. Hierarchical phrase-based MT and syntax-based MT use a grammar consisting of synchronous context-free grammar rules. However, in order to use these advanced models, data needs to be preprocessed to extract the required new information. Unfortunately, the nature of learner data, where grammatical errors are present, and its divergences from native English mean that existing NLP tools, such as sentence splitters, lemmatisers, POS taggers and parsers, do not always perform well. The majority of NLP tools are developed exclusively from high quality copy-edited text and not trained to deal with the non-standard language used by learners of English. Thus, the performance of these NLP tools may be negatively affected by grammatical errors present in the text. As shown by Napoles et al. (2016a), increasing the number of errors in a sentence decreases the accuracy of the dependency parser. Therefore, it is unclear whether using advanced TMs that require more sophisticated linguistic processing would yield better performance for GEC. We investigate the use of three different types of TMs in Section 3.3.2.

On the other hand, existing alignment tools used in SMT assume that no detailed mapping information is provided. Correction mappings (i.e. translation phrase mappings) learnt by these alignment models depend only on the size and quality of the training data. However, we observe that most words on the source side should be aligned to themselves on the target side as they do not contain any errors (i.e. self-translation). Additionally, in fully annotated learner corpora, error annotations contain detailed correction information, e.g. where and how to correct. This information is very valuable but not used by any of the alignment tools in standard SMT. Mapping information extracted directly from error annotations may give better and more accurate alignments, resulting in better correction. We investigate the use of extracted mapping information in two ways, using it as additional training data (Section 3.3.4.3) and to build a new phrase translation table directly (Section 3.3.5).

3.3 Experiments

3.3.1 Experimental set-up

The work presented in this chapter is in the context of the CoNLL-2014 shared task on grammatical error correction (see Section 2.5.2). NUCLE v3.0 provided by the shared task organisers is used as in-domain training data and results are reported on the development set, which contains 50 essays from the CoNLL-2013 shared task test set (see Section 2.5.2). The sizes of these two datasets are given in Table 3.1. System performance is evaluated in terms of $F_{0.5}$ as computed by the M^2 scorer with default settings.

SMT-based GEC systems are built using Moses, an open source toolkit for SMT developed by Koehn et al. (2007). For word alignment, we use the unsupervised alignment tool GIZA++, which is an implementation of IBM Models 1-5 and the HMM. Word alignments learnt by GIZA++ are used to extract phrase-to-phrase

Type	Dataset	Sentences	Tokens
Training set	NUCLE	57,152	1,220,257
Development set	CoNLL-2013 test	1,381	29,207

Table 3.1: CoNLL-2014 dataset sizes.

translations using heuristics. Default settings in Moses are used: 5 iterations of IBM Model 1; 0 iterations of IBM Model 2; 5 iterations of HMM; 3 iterations of IBM Model 3; 3 iterations of IBM Model 4 and 0 iterations of IBM Model 5. Bidirectional runs of GIZA++ are performed to make an alignment from erroneous to corrected sentences and another from corrected to erroneous sentences. Two word alignment files are created which are then used to derive the final word alignments. These final word alignments are the intersection of the two unidirectional alignments plus some additional alignment points from the union of the bidirectional runs. Phrase-to-phrase translations are extracted and scored based on the final word alignments. The five scores used in a phrase translation table are:

- direct and inverse phrase translation probability ($\varphi(c|e)$ and $\varphi(e|c)$):

  given a phrase pair $(\tilde{x}, \tilde{y})$, the phrase translation probability $\varphi(\tilde{x}|\tilde{y})$ is computed as

$$\varphi(\tilde{x}|\tilde{y}) = \frac{count(\tilde{x}, \tilde{y})}{count(\tilde{y})} \tag{3.31}$$

- direct and inverse lexical weighting ($lex(c|e)$ and $lex(e|c)$):

  given a phrase pair $(\tilde{x}, \tilde{y})$ and a word alignment a between $\tilde{x}$ word positions (j) and $\tilde{y}$ word positions (i), the lexical weighting $lex(\tilde{x}|\tilde{y})$ is computed as

$$lex(\tilde{x}|\tilde{y}, a) = \prod_{j=1}^{|x|} \frac{1}{|\{i|(i,j) \in a\}|} \sum_{\forall(i,j)\in a} P(\tilde{x}_j|\tilde{y}_i) \tag{3.32}$$

  where $P(\tilde{x}_j|\tilde{y}_i)$ can be estimated from a statistical word dictionary, usually IBM Model 4:

$$P(\tilde{x}_j|\tilde{y}_i) = \frac{count(\tilde{x}_j, \tilde{y}_i)}{count(\tilde{y}_i)} \tag{3.33}$$

- phrase penalty:

  constant value e, which penalises the introduction of new phrases during decoding.

A lexicalised reordering model which allows for phrase reorderings is also created during the phrase extraction phase. The LMs used during decoding are built from the corrected sentences in the learner corpus to make sure that the final system

TM	CE	ME	UE	P (%)	R (%)	$F_{0.5}$ (%)	F_1 (%)
Phrase-based	282	3,224	553	33.77	8.04	**20.60**	12.99
Factored	505	3,001	2,435	17.18	**14.40**	16.54	**15.67**
Syntax-based	129	3,377	219	**37.07**	3.68	13.17	6.69

Table 3.2: Results of using different TMs on the development set. The best results are marked in **bold**.

outputs fluent English sentences. The IRSTLM Toolkit (Federico et al., 2008) is used to build n-gram LM (up to 5-grams) with *modified Kneser-Ney smoothing*. The IRSTLM Toolkit is an open source language modelling toolkit for estimating, storing and accessing very large LMs. LM loading in IRSTLM is fast as it reduces storage and decoding memory requirements.

Data preprocessing is performed using the Natural Language Toolkit (NLTK) (Bird et al., 2009), the same toolkit used by the organisers to preprocess the NUCLE data and the CoNLL 2013 and 2014 test sets.

3.3.2 Translation models

We compare three types of TMs within the context of an all-errors correction task: phrase-based MT, factored MT and syntax-based MT. In phrase-based MT, a TM is learnt from the parallel sentences based only on the lexical representation (i.e. surface forms). Every entry in the translation table is a phrase-to-phrase mapping. Non-compositional phrases (arbitrary sequences of words) are used as translation units and local contexts are also encoded.

In factored MT, each word in the training data is represented using not only its surface form but also its POS tag and lemma. As discussed in Section 3.2, the source side of the training data may contain grammatical errors so the POS and lemma information obtained from existing NLP tools (e.g. NLTK) may be unreliable. Therefore, we only add POS and lemma factors on the target side of the training data, which consists of corrected, error-free sentences. In addition, new POS-based and lemma-based LMs are built and used during decoding.

Syntax-based MT operates at the syntax level and extracts hierarchical grammar rules from the training data. Linguistic information from both the source and target sides of the training data is needed, so the preprocessing of the erroneous data in the source cannot be avoided. Compared with phrases used in phrase-based MT and factored MT, translation grammar rules used in syntax-based MT allow long distance constraints and encode long-range contextual information.

Results using the aforementioned TMs are presented in Table 3.2. Following suggestions from Leacock et al. (2014), we report not only F-scores but also P, R and the counts from which P and R can be calculated (i.e. correct edits (CEs), missed edits (MEs) and unnecessary edits (UEs)). We can see that the factored MT system achieves the highest R but the lowest P and it proposes many more changes than the other two systems when using the small NUCLE training set. The factored MT system generalises more by learning from higher-level information like POS and

LM Dataset	Data type	Sentences	Tokens
NUCLE	learner data	57,152	1,220,257
CLC	learner data	1,965,727	29,219,128
BNC	native data	29,096,733	1,010,250,770

Table 3.3: LM dataset sizes.

lemma. It covers more errors at the cost of lower P, which results in many UEs. It yields the highest F_1, which confirmed the choice of a factored MT model for the CoNLL-2013 shared task as the evaluation was based on F_1 (Yuan and Felice, 2013). The syntax-based MT system seems to be the most conservative and the best for P, as it only makes changes to the source sentence when the probability of an error is high. The syntax-based MT system does not generalise as well as the other two systems where phrase mappings are used and only local contexts are considered. There are more constraints during translation in the syntax-based MT system, such as long distance constraints and long-range contexts. This reduces the number of UEs and lowers R. The phrase-based MT system gives the most balanced performance in terms of P and R as well as showing the best $F_{0.5}$ overall. We therefore use it in future experiments where we try to optimise $F_{0.5}$.

3.3.3 Language models

LMs are used to make sure the final output from an SMT system is fluent English. Correct and/or error corrected English sentences can be used to build LMs. Previous work has shown that adding bigger LMs based on larger corpora improves system performance (Yuan, 2013; Yuan and Felice, 2013). Therefore, apart from the target side of the parallel training data (i.e. the corrected version of NUCLE), we introduce two new corpora to build bigger LMs: the corrected version of the CLC and the written part of the British National Corpus (BNC) v1.0, which consists of texts extracted from a wide range of sources, such as newspapers, academic books, popular fiction, letters, school and university essays. The sizes of these three datasets (NUCLE, CLC and BNC) are given in Table 3.3. These new corpora are used in two ways. First, new data is added to the target side of the parallel training data to build one larger LM for decoding. Second, new data is used to build a second LM, which is then used together with the original LM built from the target side of the parallel training corpus, to ensure the bigger corpus (e.g. CLC or BNC) does not take over the overall LM and prevent corpus bias.

A set of experiments using different LMs during SMT decoding are reported in Table 3.4. We can see that all the systems with bigger LMs outperform the one using only the default NUCLE LM (#0) in terms of $F_{0.5}$, except *NUCLE&CLC* (#4). Using only one LM (#1-3) yields better P and $F_{0.5}$, while using two separate LMs (#4-6) yields better R. When building only one LM, *NUCLE+CLC* (#1) outperforms *NUCLE+BNC* (#2) and *NUCLE+CLC+BNC* (#3), suggesting that adding the CLC seems more effective than the BNC. One possible reason is that the CLC is much more similar to the NUCLE corpus than the BNC in many ways,

#	LM	CE	ME	UE	P (%)	R (%)	$F_{0.5}$ (%)
0	NUCLE (default)	282	3,224	553	33.77	8.04	20.60
1	NUCLE+CLC	359	3,147	673	34.79	10.24	**23.51**
2	NUCLE+BNC	305	3,201	576	34.62	8.70	21.69
3	NUCLE+CLC+BNC	326	3,180	569	**36.42**	9.30	23.00
4	NUCLE&CLC	547	2,959	1,918	22.19	**15.60**	20.46
5	NUCLE&BNC	513	2,993	1,720	22.97	14.63	20.62
6	NUCLE+CLC&BNC	537	2,969	1,844	22.55	15.32	20.61

Table 3.4: Results of using different LMs on the development set. '+' indicates that data is added together to build a single LM while '&' indicates that two separate LMs are built and used during decoding. The best results are marked in **bold**.

e.g. data type, topic, vocabulary, syntax. It seems that the effect of using new data for LM training in SMT-based GEC systems depends on the quality of the new data, so that the closer the new data is to the original parallel training data, the better the performance. Moreover, the quality of the data seems more important than the quantity, as the BNC is about 35 times the size of the CLC (in terms of tokens - see Table 3.3). Overall, *NUCLE+CLC+BNC* (#3) achieves the highest P, *NUCLE&CLC* (#4) yields the highest R, and *NUCLE+CLC* (#1) shows the best $F_{0.5}$, so it is used in later experiments.

3.3.4 Increasing the size of the training set

The performance of an SMT system depends on the quantity and quality of available training data (Koehn et al., 2003; Suresh, 2010; Axelrod et al., 2011). The NUCLE training set is considered too small to build good SMT systems, as previous work has shown that training on small datasets does not yield particularly high performance for SMT-based GEC (Mizumoto et al., 2012; Yuan and Felice, 2013). A few strategies have been proposed to overcome this problem. Brockett et al. (2006) transformed well-formed edited English sentences into mostly ungrammatical strings by introducing artificial mass noun errors. Similarly, Mizumoto et al. (2011) and Junczys-Dowmunt and Grundkiewicz (2014) extracted real learner examples from Lang-8. Grundkiewicz and Junczys-Dowmunt (2014) introduced the WikEd Error Corpus, which consists of sentences extracted from Wikipedia revision histories. As data collected from Lang-8 and Wikipedia revision histories may be too error-prone and noisy, an error selection process is performed (see Section 2.2).

Instead, we propose three ways to increase our training dataset size: 1) extract parallel sentences from other high-quality learner corpora; 2) generate artificial data by injecting errors into error-free English sentences; and 3) add short parallel phrases extracted from error annotations.

3.3.4.1 Adding learner data

Fully annotated learner corpora are especially valuable for GEC as they contain real learner errors and corrections made by professional annotators. Parallel sentences extracted from these annotated learner corpora can be used as SMT training data. Apart from the NUCLE training set provided by the shared task organisers, two high-quality learner corpora are used (see Section 2.3.2):

- FCE:
  - the FCE subcorpus of the CLC;
  - approximately 33,686 pairs of parallel sentences and 538,553 tokens on the target side;
- IELTS:
  - the IELTS subcorpus of the CLC;
  - approximately 54,748 pairs of parallel sentences and 1,383,245 tokens on the target side.

3.3.4.2 Adding artificial data

Fully annotated learner corpora are expensive and limited. Following previous approaches (Brockett et al., 2006), we increase the size of our training set by introducing new sentences containing artificial errors. New errors are injected into error-free English sentences based on some statistics from learner corpora. We first estimate probabilities in a learner corpus, computing the probability of each error type $P(t)$ occurring over the total number of relevant instances (e.g. noun phrases are relevant instances for article errors). During generation, $P(t)$ is uniformly distributed over all the possible choices for the error type (e.g. for articles, choices are $\{a/an, the, \phi\}$). Relevant instances are detected in the base text and changed for an alternative at random using the estimated probabilities. The probability of leaving relevant instances unchanged is $1 - P(t)$. When collecting the base text for error injection, a set of variables need to be considered, namely topic, genre, style/register, text complexity/language proficiency and L1. Two artificial datasets are created using this method on two types of base text (Felice and Yuan, 2014b):

- EVP:
  - a set of sentences from the English Vocabulary Profile (EVP) website,[1] a publicly available portion of the CLC;
  - approximately 18,830 pairs of parallel sentences and 349,343 tokens on the target side;

[1] `http://www.englishprofile.org/wordlists`

- Wiki:
  - a set of 494 Wikipedia articles chosen based on keywords in the NUCLE corpus, to ensure compatibility of topics;[2]
  - approximately 54,693 pairs of parallel sentences and 1,120,697 tokens on the target side.

3.3.4.3 Adding short parallel phrases

As noted by Yuan (2013) and Yuan and Felice (2013), we also notice alignment errors, suggesting that the unsupervised alignment tool used in SMT is not reliable enough to learn useful mappings from a relatively small parallel corpus. These alignment errors result in missed or unnecessary corrections in the final SMT system. As discussed at the beginning of this chapter (see Section 3.2), error annotations often encode useful alignment information which is not used by any alignment tool in SMT. In order to exploit it, short phrase alignments that include up to 7 tokens per side within one sentence boundary and involve corrections for each error are extracted from learner corpora. Different errors are treated the same regardless of their type. An example is given below:

Example 3.4. *In */the modern digital world , electronic products are widely used in daily *lives/life such as smart phones , computers and etc .*[3]

We can easily get the following word mappings from the annotation:

NULL → the
lives → life

Short phrases containing the erroneous word (or phrase in some cases) and its neighbouring tokens within a 7-token window are extracted, such as:

modern → the modern
modern digital → the modern digital
modern digital world → the modern digital world
daily lives → daily life
in daily lives → in daily life

A full list of the extracted phrase alignments for the illustrated article and noun number errors are presented in Figure 3.3.

These extracted short phrases can then be used as new training examples. Phrase alignments are added into the training set and existing alignment tools are then used to create a phrase table from the new augmented training set.

We extract these new phrases from NUCLE. To give frequent phrase mappings higher probability than infrequent ones, we try keeping all their occurrences. Two versions of the phrase-level training data are created, with and without duplicates:

[2]We choose an initial set of 50 Wikipedia articles based on keywords in the NUCLE training data and proceed to collect related articles by following hyperlinks in their 'See also' section. We retrieve a total of 494 articles which are later preprocessed to remove Wikicode tags.

[3]This sentence is extracted from the NUCLE corpus. The use of 'and etc' is wrong, but this error was not annotated in the original corpus (i.e. it constitutes an annotation error).

missing determiner 'the': *NULL* → *the*
modern → the modern
modern digital → the modern digital
modern digital world → the modern digital world
modern digital world , → the modern digital world ,
modern digital world , electronic → the modern digital world , electronic
modern digital world , electronic products → the modern digital world , electronic products
In → In the
In modern → In the modern
In modern digital → In the modern digital
In modern digital world → In the modern digital world
In modern digital world , → In the modern digital world ,
In modern digital world , electronic → In the modern digital world , electronic
noun number error: *lives* → *life*
lives → life
lives such → life such
lives such as → life such as
lives such as smart → life such as smart
lives such as smart phones → life such as smart phones
lives such as smart phones , → life such as smart phones ,
lives such as smart phones , computers → life such as smart phones , computers
daily lives → daily life
daily lives such → daily life such
daily lives such as → daily life such as
daily lives such as smart → daily life such as smart
daily lives such as smart phones → daily life such as smart phones
daily lives such as smart phones , → daily life such as smart phones ,
in daily lives → in daily life
in daily lives such → in daily life such
in daily lives such as → in daily life such as
in daily lives such as smart → in daily life such as smart
in daily lives such as smart phones → in daily life such as smart phones
used in daily lives → used in daily life
used in daily lives such → used in daily life such
used in daily lives such as → used in daily life such as
used in daily lives such as smart → used in daily life such as smart
widely used in daily lives → widely used in daily life
widely used in daily lives such → widely used in daily life such
widely used in daily lives such as → widely used in daily life such as
are widely used in daily lives → are widely used in daily life
are widely used in daily lives such → are widely used in daily life such
products are widely used in daily lives → products are widely used in daily life

Figure 3.3: Phrase alignments extracted from the sentence in Example 3.4.

- $\text{NUCLE}_{\text{phrase\_1}}$:
  - NUCLE phrase version 1, where identical phrase pairs are kept;
  - approximately 606,679 pairs of parallel phrases and 2,776,181 tokens on the target side;

- $\text{NUCLE}_{\text{phrase\_2}}$:
  - NUCLE phrase version 2, where identical phrase pairs are removed;
  - approximately 570,798 pairs of parallel phrases and 2,697,919 tokens on the target side.

3.3.4.4 Results

In our previous experiments, we used heuristics to extract phrases from the word alignments learnt by GIZA++. Compared with the heuristic phrase extraction method, Pialign is an unsupervised model for joint phrase alignment and extraction using non-parametric Bayesian methods and Inversion Transduction Grammars (ITGs) (Neubig et al., 2011). Alignments are obtained through Bayesian learning of ITG trees (Wu, 1997), where each pair of parallel sentences is represented as a tree of aligned phrases and binary reordering operations. We compare phrase tables constructed using these two phrase extraction methods. New learner datasets (FCE and IELTS) and artificial datasets (EVP and Wiki) are added to the NUCLE training set incrementally. We add the two versions of the NUCLE phrase data ($\text{NUCLE}_{\text{phrase\_1}}$ and $\text{NUCLE}_{\text{phrase\_2}}$) to a system trained only on NUCLE as well as to our best overall system. Results are presented in Table 3.5.

We can see that adding parallel sentences extracted from other annotated learner corpora (FCE and IELTS) yields a consistent improvement in system performance; that is, the more learner data, the better. The artificial data generated from learner text (EVP) seems helpful when building SMT systems with both alignment methods, but the data generated from native text (Wiki) is only useful when building systems with GIZA++. Systems using Pialign yield consistently better P while those using GIZA++ yield consistently better R. Before adding the phrase-level data, the best systems in terms of $F_{0.5}$ are the ones trained on *NUCLE+FCE+IELTS+EVP+Wiki* (for GIZA++) and *NUCLE+FCE+IELTS+EVP* (for Pialign).

The advantage of using high-quality learner corpora like FCE and IELTS is that they contain real examples produced by learners. The parallel data extracted from them is close enough to NUCLE so it results in improvement. Unfortunately, this kind of data is limited as new learner corpora are very expensive to build. The use of artificial data overcomes this problem as we can make use of unlimited native data. Artificial data is easy to generate and can be tailored to our needs. However, the effectiveness of genuine and artificial data is not the same. Our results show that the learner data extracted directly from the FCE and IELTS datasets is more useful than the artificially generated data. For the artificial data, the choice of the error-free base text is important. We can see that the artificial data generated from the error-free learner text (EVP) yields better performance than the one generated from the native text (Wiki). This suggests that we can exploit error-free text written by learners if error-annotated text is not available or reliable. It also suggests that we should choose native base text that resembles learner data for error injection.

Using short parallel phrases extracted from NUCLE as additional training data yields a consistent improvement in $F_{0.5}$, while R improves at the cost of P. If

Alignment	Training data	CE	ME	UE	P (%)	R (%)	$F_{0.5}$ (%)
GIZA++	NUCLE	359	3,147	673	**34.79**	10.24	23.51
	NUCLE+FCE	400	3,106	793	33.53	11.41	24.16
	NUCLE+FCE+IELTS	504	3,002	1,106	31.30	14.38	25.34
	NUCLE+FCE+IELTS+EVP	494	3,012	981	33.49	14.09	26.26
	NUCLE+FCE+IELTS+EVP+Wiki	505	3,001	977	34.08	14.40	26.76
	NUCLE+$NUCLE_{phrase\_1}$	541	2,965	1,386	28.07	15.43	24.12
	NUCLE+$NUCLE_{phrase\_2}$	524	2,982	1,326	28.32	14.95	24.02
	NUCLE+FCE+IELTS+EVP+Wiki+$NUCLE_{phrase\_1}$	607	2,899	1,295	31.91	**17.31**	**27.31**
	NUCLE+FCE+IELTS+EVP+Wiki+$NUCLE_{phrase\_2}$	604	2,902	1,284	31.99	17.23	**27.31**
Pialign	NUCLE	160	3,346	183	46.65	4.56	16.40
	NUCLE+FCE	201	3,305	243	45.27	5.73	19.03
	NUCLE+FCE+IELTS	327	3,179	348	48.44	9.33	26.35
	NUCLE+FCE+IELTS+EVP	331	3,175	359	47.97	9.44	26.41
	NUCLE+FCE+IELTS+EVP+Wiki	295	3,211	306	**49.08**	8.41	24.96
	NUCLE+$NUCLE_{phrase\_1}$	462	3,044	963	32.42	13.18	25.09
	NUCLE+$NUCLE_{phrase\_2}$	383	3,123	794	32.54	10.92	23.31
	NUCLE+FCE+IELTS+EVP+$NUCLE_{phrase\_1}$	471	3,035	775	37.80	**13.43**	27.74
	NUCLE+FCE+IELTS+EVP+$NUCLE_{phrase\_2}$	470	3,036	751	38.49	13.41	**28.01**

Table 3.5: Results of adding more training data on the development set. The best results using each alignment tool are marked in **bold**.

we add these phrase pairs to the systems trained only on NUCLE, leaving duplicate phrase pairs ($NUCLE_{phrase\_1}$) outperforms removing them ($NUCLE_{phrase\_2}$). However, when we add them to the current best systems per alignment method, *NUCLE+FCE+IELTS+EVP+Wiki* (GIZA++) and *NUCLE+FCE+IELTS+EVP* (Pialign), $NUCLE_{phrase\_2}$ seems more helpful than $NUCLE_{phrase\_1}$. This extracted phrase-level training data is used to boost the probability of phrase alignments that involve corrections, so as to improve R. Yet, our extracted phrases are more useful with Pialign than GIZA++. Overall, the best system in terms of $F_{0.5}$ is the one trained on *NUCLE+FCE+IELTS+EVP+*$NUCLE_{phrase\_2}$ and aligned with Pialign.

We also extract phrases from the FCE dataset, but results show that adding them to the training set is not helpful on NUCLE. This might be caused by the differences between these two learner corpora, such as the different sources of data and annotation schemes.

3.3.5 A new method for building a phrase table

In phrase-based MT, a phrase translation table is learnt from the parallel training data. Phrase mappings in the table are then used by the decoder as translation units. Rather than learning phrase mappings from the parallel sentences using unsupervised alignment tools like GIZA++ and Pialign, we propose a new method to create a new phrase table using phrase pairs extracted from error annotations directly. By dispensing with unsupervised alignment, our method saves much time and effort. Phrase pairs extracted from error annotations are used as phrase mappings in the new phrase table. The mappings in the new table include the same phrase translation probabilities, lexical weighting probabilities and phrase penalty score used by GIZA++ and Pialign (see Equation 3.31 and 3.32).

In error correction, most words translate into themselves as they are usually correct. We also notice that errors are often similar to their correct forms, such

as noun number errors (*one *years/year ago*) or word form errors (*the sense of *guilty/guilt*). Therefore, a new type of feature based on Levenshtein distance is introduced to limit the changes made by the SMT system. Levenshtein distance is a string metric for measuring the difference between two sequences. It calculates the minimum number of edits (i.e. insertions, deletions or substitutions) required to change one sequence into another. Given a phrase pair $(\tilde{x}, \tilde{y})$, the Levenshtein distance feature $score_{LD}(\tilde{x}, \tilde{y})$ is defined as

$$score_{LD}(\tilde{x}, \tilde{y}) = \frac{max\{N(\tilde{x}), N(\tilde{y})\} - LD(\tilde{x}, \tilde{y})}{max\{N(\tilde{x}), N(\tilde{y})\}} \tag{3.34}$$

where $LD(\tilde{x}, \tilde{y})$ is the Levenshtein distance between $\tilde{x}$ and $\tilde{y}$, and $N(\cdot)$ is the sequence length.

Matching can be done at the word or character level, as shown below:

Example 3.5. For the following phrase mapping:
I am so exciting → I am so excited

The word-level Levenshtein distance feature score is:

I	am	so	exciting
I	am	so	excited
M	M	M	S

$score_{LD_w} = \frac{4-1}{4} = 0.75$

The character-level Levenshtein distance feature score is:

I	<s>	a	m	<s>	s	o	<s>	e	x	c	i	t	i	n	g
I	<s>	a	m	<s>	s	o	<s>	e	x	c	i	t	e	d	
M	M	M	M	M	M	M	M	M	M	M	M	M	S	S	D

$score_{LD_c} = \frac{16-3}{16} = 0.8125$

where M: match, S: substitution, D: deletion

As we see in this example, character-level Levenshtein distance captures words with identical stems and unigram paraphrases. Character-level Levenshtein distance is assumed to work better than the word-level one, as it can provide additional information for word form errors, noun number errors and contextual spelling errors (e.g. 'their' and 'there').

When building a phrase table using our new method, Levenshtein distance is used to measure the difference between the source and target phrases. A series of preliminary experiments is first undertaken to compare the word-level and character-level Levenshtein distance features. Results confirm our hypothesis that the use of the character-level Levenshtein distance feature yields better performance (see Table 3.6), which is why we choose it for inclusion in our phrase table.

We want to compare our method with GIZA++ and Pialign for building phrase translation tables. Two sets of experiments are performed. We first use only the NUCLE data provided by the shared task organisers, assuming that it is the only training data available; and then we compare the three methods under their most favourable conditions allowing them to use any of the sentence-level training data introduced in Section 3.3.4. Results are presented in Table 3.7. We can see that

Feature	P	R	$F_{0.5}$
phrase translation probabilities, lexical weighting probabilities, phrase penalty	26.57	13.93	22.48
+ word-level LD	26.94	13.74	22.60
+ character-level LD	27.96	13.61	23.09

Table 3.6: Results of using our new phrase table method with different features (in percentages). Systems are trained on the NUCLE training set.

Alignment	Training data	P	R	$F_{0.5}$
GIZA++	NUCLE	34.79	10.24	23.51
	NUCLE+FCE+IELTS+EVP+Wiki	34.08	14.40	26.76
Pialign	NUCLE	46.65	4.56	16.40
	NUCLE+FCE+IELTS+EVP	47.97	9.44	26.41
Our method	NUCLE	27.96	13.61	23.09
	NUCLE+FCE+IELTS+EVP	26.79	16.00	23.61

Table 3.7: Results of using different alignment methods on the development set (in percentages).

the SMT system using our phrase table method is competitive with the one using GIZA++ in terms of $F_{0.5}$, and they both outperform the system using Pialign by a large margin when using only the NUCLE corpus. However, adding more training data does not help our method as much as GIZA++ and Pialign. Adding additional training examples and using new phrases extracted from other learner corpora only yield a 0.52 increase in $F_{0.5}$ for our method, compared with a 3.25 increase for GIZA++ and a 10.01 increase for Pialign. One possible explanation is that phrases extracted from other learner corpora are different from those extracted from NUCLE (as we have shown in Section 3.3.4.4), so adding phrase pairs extracted from the FCE, IELTS and EVP datasets is not helpful for NUCLE. However, using our method achieves the highest R (at the cost of P, though), which again shows that phrase pairs extracted from error annotations can help achieve better R (see Section 3.3.4.4). In terms of training time, our method is the fastest while Pialign is the slowest.

Even though our method does not result in the best SMT system, the use of the character-level Levenshtein distance feature seems beneficial for SMT systems (see Table 3.6) so we further explore how this interacts with other alignment tools (e.g. GIZA++ or Pialign) in Section 3.4.1 and 4.4.2.

3.3.6 Forced decoding for phrase table filtering

As described in Section 3.1.4, an SMT decoder searches the space of possible translations (candidates) and outputs the highest scoring one. A phrase translation table learnt from parallel training data is used by the decoder to translate source sentences. However, not all the phrase mappings in the table are useful or reliable. Only some of them produce good translations while the rest may result in translation errors. For this reason, we propose a method for filtering the translation table.

Forced decoding in SMT is used to force the decoder to output only the candidate that is the same as the gold-standard reference. Phrase alignments that are used by the decoder to produce the expected translations are considered to be useful and therefore valuable for phrase table filtering.

In order to collect useful phrase alignments, we adapt the 4-fold cross-validation scheme used by Yuan and Felice (2013). The training set for each run always includes the full learner data, artificial data and phrase-level data introduced in Section 3.3.4, but only 3/4 of NUCLE (in-domain training data), leaving the remaining fourth chunk for testing with forced decoding. This training method allows us to concentrate on the performance of the system on the NUCLE data. After each run, we collect a list of phrase alignments that are used by the decoder at least once during forced decoding. Phrase alignments collected from each run are then combined into one big list. Finally, we learn a new phrase translation table from the full training data and filter it using the list of useful phrase alignments. More specifically, for every phrase mapping in the translation table, we check whether it is also in the phrase alignment list. For phrase mappings not in the list, we probably do not want to simply remove them from the table as they might be useful to translate some new test examples. Instead, we can either decrease their probabilities with a 'scale-down' factor f_d, where $0 < f_d < 1$; or increase the probabilities of the other phrase mappings with a 'scale-up' factor f_u, where $f_u > 1$.

We evaluate our filtering method by comparing SMT systems with and without phrase table filtering. We experiment with tables produced by GIZA++ and Pialign. In Table 3.8, *No filtering* is the best SMT system using GIZA++, which is trained on *NUCLE+FCE+IELTS+EVP+Wiki+*$NUCLE_{phrase\_2}$ (see Table 3.5). *Removing* uses a translation table where phrase mappings that are not in the list are removed. The rest are systems using different scaling factors. Similarly, in Table 3.9, *No filtering* is the best SMT system trained on *NUCLE+FCE+IELTS+EVP+*$NUCLE_{phrase\_2}$ and aligned with Pialign. We can see that *Removing* does not yield good performance. Decreasing the probabilities for phrases that are not in the list (scaling down) shows a consistent improvement in P over systems with no filtering, while increasing the probabilities of phrase mappings that are in the list (scaling up) yields a consistent improvement in R. As we increase f_u, R keeps increasing and P keeps decreasing (see Figure 3.4 and 3.5). However, not all the filtered systems yield better performance in terms of $F_{0.5}$. Filtering with a scale-down factor of $f_d = 0.5$ seems the most effective, as it outperforms systems with no filtering and achieves the highest $F_{0.5}$ for both alignment methods.

Filtering		CE	ME	UE	P (%)	R (%)	$F_{0.5}$ (%)
No filtering		604	2,902	1,284	31.99	17.23	27.31
Removing		313	3,193	613	**33.80**	8.93	21.71
Scaling down	$f_d = 0.1$	364	3,142	701	**34.18**	10.38	23.44
	$f_d = 0.3$	479	3,027	894	**34.89**	13.66	26.62
	$f_d = 0.5$	521	2,985	978	**34.76**	14.86	**27.42**
	$f_d = 0.7$	553	2,953	1,101	**33.43**	15.77	**27.32**
	$f_d = 0.9$	589	2,917	1,237	**32.26**	16.80	27.24
Scaling up	$f_u = 1.1$	610	2,896	1,309	31.79	**17.40**	27.28
	$f_u = 1.5$	631	2,875	1,413	30.87	**18.00**	27.01
	$f_u = 2.0$	648	2,858	1,519	29.90	**18.48**	26.61
	$f_u = 2.5$	664	2,842	1,624	29.02	**18.94**	26.23
	$f_u = 3.0$	682	2,824	1,720	28.39	**19.45**	26.00

Table 3.8: Filtering results on the development set using GIZA++ tables. Improvements over the *No filtering* system are marked in **bold**.

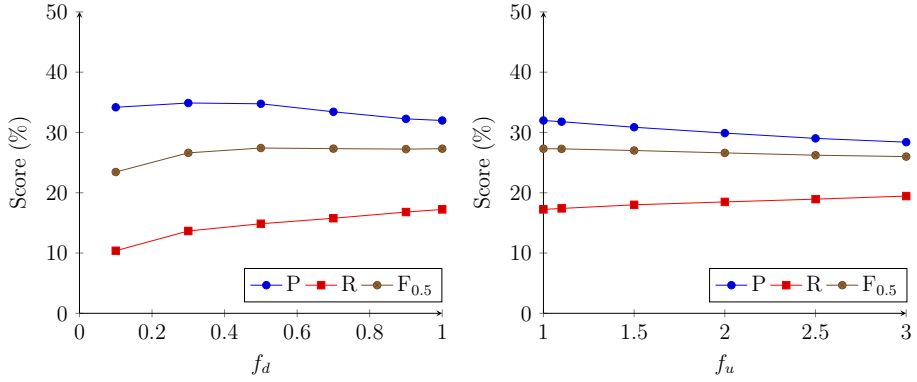
Filtering		CE	ME	UE	P (%)	R (%)	$F_{0.5}$ (%)
No filtering		470	3,036	751	38.49	13.41	28.01
Removing		273	3,233	419	**39.45**	7.79	21.76
Scaling down	$f_d = 0.1$	291	3,215	436	**40.03**	8.30	22.68
	$f_d = 0.3$	361	3,145	519	**41.02**	10.30	25.69
	$f_d = 0.5$	420	3,086	560	**42.86**	11.98	**28.28**
	$f_d = 0.7$	438	3,068	653	**40.15**	12.49	27.83
	$f_d = 0.9$	458	3,048	726	**38.68**	13.06	27.78
Scaling up	$f_u = 1.1$	476	3,030	773	38.11	**13.58**	27.99
	$f_u = 1.5$	485	3,021	807	37.54	**13.83**	27.96
	$f_u = 2.0$	499	3,007	896	35.77	**14.23**	27.46
	$f_u = 2.5$	509	2,997	1,021	33.27	**14.52**	26.44
	$f_u = 3.0$	521	2,985	1,080	32.54	**14.86**	26.29

Table 3.9: Filtering results on the development set using Pialign tables. Improvements over the *No filtering* system are marked in **bold**.

3.4 An end-to-end SMT-based GEC system

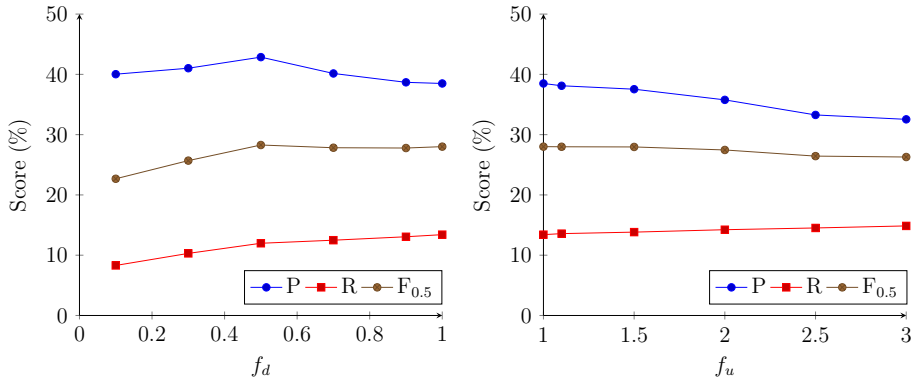
3.4.1 System performance

The CoNLL-2014 shared task organisers made NUCLE v3.1 available six days before the release of the CoNLL-2014 test data. Compared with NUCLE v3.0 used in our experiments, the new version includes some changes, such as the removal of duplicate annotations, fixed end-of-paragraph annotations and corrected annotation mistakes. Given these changes, we build a new phrase-based SMT system using NUCLE v3.1 and re-evaluate system performance on the development set. Based on our findings



(a) Scaling down **(b)** Scaling up

Figure 3.4: Phrase table filtering (GIZA++).



(a) Scaling down **(b)** Scaling up

Figure 3.5: Phrase table filtering (Pialign).

in Section 3.3, we use Pialign for word alignment and add several modifications to the final system:[4]

- NUCLE:

  uses NUCLE v3.1 released by the shared task organisers as in-domain training data;

- LM:

  builds a bigger LM by adding the corrected version of the CLC for decoding - see Section 3.3.3;

- FCE:

  incorporates all sentences in the FCE dataset - see Section 3.3.4.1;

[4]The phrase table filtering discussed in Section 3.3.6 was not implemented in the final SMT system because of limited time.

Setting	P	R	$F_{0.5}$
NUCLE	46.63	3.04	12.06
NUCLE+LM	46.70	4.61	16.52
NUCLE+LM+FCE	45.32	5.78	19.14
NUCLE+LM+FCE+IELTS	48.50	9.36	26.41
NUCLE+LM+FCE+IELTS+EVP	47.98	9.49	26.49
NUCLE+LM+FCE+IELTS+EVP+$\text{NUCLE}_{\text{phrase\_2}}$	38.50	13.46	28.06
NUCLE+LM+FCE+IELTS+EVP+$\text{NUCLE}_{\text{phrase\_2}}$+LD	39.58	13.23	28.30

Table 3.10: Incremental results on the development set (in percentages).

- IELTS:

  incorporates all sentences in the IELTS dataset - see Section 3.3.4.1;

- EVP:

  adds artificial parallel training data generated from the EVP corpus - see Section 3.3.4.2;

- $\text{NUCLE}_{\text{phrase\_2}}$:

  uses short phrase alignments extracted from the error annotations in NUCLE v3.1 - see Section 3.3.4.3;

- LD:

  limits edit distance by adding character-level Levenshtein distance as a new feature - see Section 3.3.5.[5]

A detailed analysis of individual modifications used in the final SMT system is presented in Table 3.10. P, R and $F_{0.5}$ computed by the M^2 scorer are reported for each variation of the SMT system. Compared with results reported in Table 3.5, systems trained with NUCLE v3.1 achieve higher scores on the development set. All modifications improve system performance in terms of $F_{0.5}$. Introducing character-level Levenshtein distance helps improve $F_{0.5}$. By adding the development set into the final training set, a similar SMT system was built and used for the CoNLL-2014 shared task (see Section 3.5).

3.4.2 Error analysis

In order to better understand the performance of the final SMT-based GEC system, we perform a detailed error analysis. This helps us understand the strengths and weaknesses of the system, as well as identify new areas for future improvement.

Since our system handles various types of errors, we are interested in studying performance by type. In order to compute performance by type, we need to know

[5]As using our phrase table does not result in the best SMT system, we investigate the use of the Levenshtein distance feature on top of the current best SMT system built using Pialign.

the error type for every gold-standard and system correction. For the former, error type information is encoded in the error annotations; however for the latter, error type information is not available, since the SMT system only outputs the corrected sentences with no indication of error type. In order to estimate error types for system corrections, we apply our type estimation strategy proposed in Felice et al. (2014), where a set of heuristic rules are defined based on common patterns observed in the NUCLE corpus. Our automatic typing method analyses the differences in word forms and POS tags between the original phrases and a system's proposed corrections, and assigns error types using pre-defined rules. The estimation accuracy is around 70% on the development set, which is considered to be acceptable for our purpose. As noted by Ng et al. (2014), predicting an appropriate error type for a system edit/correction out of the 28 types used in NUCLE can be error-prone and tricky.

Type-specific performance is reported in Table 3.11. Although per-type UE, P and $F_{0.5}$ are estimated and therefore not completely accurate, they can still provide valuable insights, at least at a coarse level. The following sections discuss our main findings.

3.4.2.1 Type performance

According to Table 3.11, our SMT system achieves the best performance for types *Wform* (word form), *Mec* (punctuation, capitalisation, spelling, typos), *Nn* (noun number) and *ArtOrDet* (article or determiner), which add up to 43.61% of the errors in the development set. Some successful corrections made by the system are shown below:

Example 3.6. *ArtOrDet* and *Nn*:

Original sentence	*I think this application is necessary and provides **the** society a lot of **benefit**.*
Smt output	*I think this application is necessary and provides society a lot of **benefits**.*
Gold standard	*I think this application is necessary and provides society a lot of **benefits**.*

Example 3.7. *Mec*:

Original sentence	*In **summery**, surveillance technology ...*
Smt output	*In **summary**, surveillance technology ...*
Gold standard	*In **summary**, surveillance technology ...*

Example 3.8. *Wform*:

Original sentence	*... they begin to **loss** their memory ...*
Smt output	*... they begin to **lose** their memory ...*
Gold standard	*... they begin to **lose** their memory ...*

Example 3.9. *ArtOrDet*:

Original sentence	*In such situation, individuals will lose ...*
Smt output	*In such **a** situation, individuals will lose ...*
Gold standard	*In such **a** situation, individuals will lose ...*

Error type	Description	CE	ME				UE	P (%)	R (%)	$F_{0.5}$ (%)
			Decoding		OOV					
			No.	Prop.	No.	Prop.				
ArtOrDet	Article or determiner	145	450	83%	93	17%	148	49.49	21.08	38.98
Cit	Citation	0	2	33%	4	67%	0	-	0.00	0.00
Mec	Punctuation, capitalisation, spelling, typos	33	90	60%	59	40%	20	62.26	18.13	41.88
Nn	Noun number	98	172	58%	124	42%	96	50.52	24.87	41.88
Npos	Noun possessive	2	13	48%	14	52%	21	8.70	6.90	8.26
Others	Other errors	2	10	30%	23	70%	6	25.00	5.71	14.93
Pform	Pronoun form	1	19	76%	6	24%	10	9.09	3.85	7.14
Pref	Pronoun reference	1	17	45%	21	55%	2	33.33	2.56	9.80
Prep	Preposition	29	169	60%	112	40%	47	38.16	9.35	23.62
Reordering	Reordering	0	0	-	0	-	13	0.00	-	0.00
Rloc-	Local redundancy	10	25	21%	93	79%	14	41.67	7.81	22.32
SVA	Subject-verb agreement	12	93	88%	13	12%	24	33.33	10.17	22.90
Sfrag	Fragment	0	0	0%	4	100%	0	-	0.00	0.00
Smod	Dangling modifier	0	1	6%	15	94%	0	-	0.00	0.00
Spar	Parallelism	2	16	50%	16	50%	0	100.00	5.88	23.81
Srun	Runons, comma splice	0	39	71%	16	29%	8	0.00	0.00	0.00
Ssub	Subordinate clause	3	28	41%	40	59%	6	33.33	4.23	14.02
Trans	Link words/phrases	5	66	49%	70	51%	18	21.74	3.55	10.73
Um	Unclear meaning	0	1	3%	33	97%	0	-	0.00	0.00
V0	Missing verb	1	6	35%	11	65%	3	25.00	5.56	14.71
Vform	Verb form	19	63	64%	36	36%	39	32.76	16.10	27.14
Vm	Verb modal	8	46	53%	41	47%	8	50.00	8.42	25.16
Vt	Verb tense	13	102	72%	40	28%	22	37.14	8.39	22.03
Woadv	Adverb/adjective position	0	0	0%	12	100%	0	-	0.00	0.00
Woinc	Incorrect sentence form	2	3	9%	32	91%	50	3.85	5.41	4.08
Wa	Acronyms	0	0	0%	5	100%	1	0.00	0.00	0.00
Wci	Wrong collocation/idiom	15	75	18%	338	82%	98	13.27	3.05	8.52
Wform	Word form	51	94	54%	81	46%	36	58.62	22.57	44.43
Wtone	Tone	0	8	62%	5	38%	0	-	0.00	0.00
TOTAL	-	452	1,608	54%	1,357	46%	690	39.58	13.23	28.30

Table 3.11: Type-specific performance of the SMT system on the development set.

After an analysis of these four types of errors, we learn that the SMT system is particularly good at correcting errors that:

1. Have more training examples.

   These four types of errors are some of the most frequent errors made by learners of English: *ArtOrDet* (ranked *1st* in the NUCLE corpus), *Nn* (*4th*), *Mec* (*6th*) and *Wform* (*8th*). There are many more training examples containing corrections for these errors than the rest, so the SMT system is more likely to learn reliable correction mappings. In addition, the relatively large proportions of repeated corrections in these four error types make it easier for the system to detect and correct those errors. This frequent repetition is not observed for other common learner errors like *Wci* (wrong collocation/idiom, *2nd*), *Rloc-*

(local redundancy, *3rd*) and *Vt* (verb tense, *5th*), partly explaining why the system does not yield good performance for them.

2. Involve changes of only one or a few words.

   Corrections for these four types of errors are mostly insertions, deletions or replacements of only one or a few words. Several short alignments are learnt and used effectively by the SMT system, such as *internet* → *Internet*, *lifes* → *lives* and *over crowded* → *overcrowded*.

3. Depend on local context.

   The phrase-based TM (where phrases contain up to 7 tokens) and the n-gram LM used in the SMT system are more likely to capture short-range context than long-range context. Therefore, the SMT system is more effective at correcting errors that require only local context information than those that depend on long-range contexts.

The SMT system yields poor performance for *Wci*: an $F_{0.5}$ score of 8.52%, a P score of 13.27%, and a R score of 3.05%. A closer inspection of the missed *Wci* errors shows that most of them are new to the system and have not been seen in the training set. Learners of English are so creative that it is not possible to collect all erroneous collocations, especially for open-class words like adjectives. Nevertheless, our system makes good changes to some incorrect phrases used by learners. The following example shows a successful correction:

Example 3.10. *Wci*:

ORIGINAL SENTENCE	*People's **life quality** has been better now ...*
SMT OUTPUT	*People's **quality of life** has been better now ...*
GOLD STANDARD	*People's **quality of life** has been better now ...*

In other cases, our system seems to do a good job despite the gold-standard annotation:

Example 3.11. *Wci*:

ORIGINAL SENTENCE	*... not only inefficient in improving our **life quality** ...*
SMT OUTPUT	*... not only inefficient in improving our **quality of life** ...*
GOLD STANDARD	*... not only inefficient in improving our **life quality** ...*

We observe that almost all the corrected *Rloc-* errors are deletions of only one word. Those involving deletions of two or more words are not handled well by the SMT system, as in the following example:

Example 3.12. *Rloc-*:

ORIGINAL SENTENCE	*Tracking people using surveillance technology can offer better security **of people's life**.*
SMT OUTPUT	*Tracking people using surveillance technology can offer better security **of people's life**.*
GOLD STANDARD	*Tracking people using surveillance technology can offer better security.*

For different types of agreement errors like *Vt* and *SVA*, the SMT system does not perform well when the useful information is far from the error, e.g. if the clue is not close to the verb in question for *Vt* or the verb and its subject are far apart for *SVA*. Consider the following examples:

Example 3.13. *Vt*:

ORIGINAL SENTENCE	*This could lead to psychological implications and thus **causing** many tremendous social effects ...*
SMT OUTPUT	*This could lead to psychological implications and thus **causing** many tremendous social effects ...*
GOLD STANDARD	*This could lead to psychological implications and thus **cause** many tremendous social effects ...*

In this example, the clue to change 'causing' to 'cause' is 'could lead', which is at the beginning of the sentence. As this information is far from the verb in question, the SMT system does not detect the error.

Example 3.14. *SVA*:

ORIGINAL SENTENCE	*People **needs** a safe environment to live in, and also **needs** a private environment to stay independent.*
SMT OUTPUT	*People **need** a safe environment to live in, and also **needs** a private environment to stay independent.*
GOLD STANDARD	*People **need** a safe environment to live in, and also **need** a private environment to stay independent.*

In this sentence, both verbs 'needs' should be changed to 'need' because they share the same subject 'People'. Our SMT system successfully detects the first error, as the first 'needs' is just next to the subject 'People'. However, it fails to detect the second error as the second 'needs' is so far from the subject.

Zero scores for P, R and $F_{0.5}$ are observed for *Cit* (citation), *Reordering*, *Sfrag* (fragment), *Smod* (dangling modifier), *Srun* (runons, comma splice), *Um* (unclear meaning), *Woadv* (adverb/adjective position), *Wa* (acronyms) and *Wtone* (tone), suggesting that the SMT system is unable to correct these types of errors. However, these results may not be truly representative as some of the error types only account for small fractions of the development set (e.g. 0.18% for *Cit*, 0.12% for *Sfrag*) and are usually too specific to particular pairs of sentences to extract general correction mappings.

3.4.2.2 Sequential errors

One of our motivations for using the SMT framework for a general error correction task is that, SMT-based GEC systems have the potential to correct multiple errors at the same time. Our analysis reveals a number of cases where the SMT system corrects two or more errors in one sentence, namely 3 cases where four errors in one sentence are corrected, 18 cases where three errors are corrected, and 59 cases where two errors are corrected. These results confirm that the SMT system is able to correct multiple types of errors as well as interacting errors simultaneously. Consider the following examples:

Example 3.15. Multiple errors:

Original sentence	*Nevertheless, provision of more medicine and better **equipments** as well as more medical **centre is** necessary to accommodate **these elders**.*
Smt output	*Nevertheless, provision of more medicine and better **equipment** as well as more medical **centres is** necessary to accommodate **the elderly**.*
Gold standard	*Nevertheless, provision of more medicine and better **equipment** as well as more medical **centres are** necessary to accommodate **the elderly**.*

The SMT system successfully detects and corrects all the errors in the sentence: *equipments* → *equipment* (*Nn*), *centre* → *centres* (*Nn*), *these* → *the* (*ArtOrDet*), and *elders* → *elderly* (*Wform*). The last two are interacting errors, as the correction of one requires the correction of the other. The only mismatch between our system's output and the gold standard is the missed *SVA*: *is* → *are*. However, we believe this is an annotation mistake: the verb 'is' should not be changed to 'are' because its subject is 'provision'. This shows that the annotation in the NUCLE corpus is not always reliable, and therefore true system performance is underestimated.

Example 3.16. Interacting errors:

Original sentence	*And all the **evidences have** been collected ...*
Smt output	*And all the **evidence has** been collected ...*
Gold standard	*And all the **evidence has** been collected ...*

Here, two interacting errors are corrected: the change from 'evidences' to 'evidence' (*Nn*) and the change from 'have' to 'has' (*SVA*).

When looking at the phrase alignment mappings used by the SMT decoder, we notice that it generally uses two or more short alignments where each targets one error, rather than a long alignment containing corrections for all the errors.

3.4.2.3 Missed errors

MEs in Table 3.11 result in low R. We observe that many missed errors are successfully corrected in the SMT n-best list. However, the highest-ranked candidate selected by the decoder is not always the best correction. This suggests that these missed errors are essentially decoding errors, as the decoder fails to choose better candidates with more corrections. For example:

Example 3.17. Missed *ArtOrDet*:

Original sentence	*The result is increasing size of population.*
Smt output	
1st	*The result is increasing size of **the** population.*
3rd	*The result is **the** increasing size of **the** population.*
Gold standard	*The result is **the** increasing size of **the** population.*

Example 3.18. Missed *Vt* and *Nn*:

Original sentence	*Initially, surveillance technology such as RFID **is** implanted into the **body** of animals.*
Smt output	
1st	*Initially, surveillance technology such as RFID **is** implanted into the **body** of animals.*
3rd	*Initially, surveillance technology such as RFID **was** implanted into the **body** of animals.*
6th	*Initially, surveillance technology such as RFID **is** implanted into the **bodies** of animals.*
7th	*Initially, surveillance technology such as RFID **was** implanted into the **bodies** of animals.*
Gold standard	*Initially, surveillance technology such as RFID **was** implanted into the **bodies** of animals.*

Some errors are new to the system as they have not been seen in the training data. Since our phrase-based SMT system is trained on surface forms, it is unaware of syntactic structures and cannot use correction mappings of the form *NN* → *NNS*.[6] Instead, it has to have seen the exact lexical pair (e.g. *movie* → *movies*) in the training data. These out-of-vocabulary (OOV) errors are missed by the system because the required correction mappings are not included in the phrase translation table learnt from the training data.

We thus categorise errors missed by the SMT system into two groups based on the cause: 1) decoding errors; and 2) OOV errors. In order to identify the cause of every missed error, we perform forced decoding. As described in Section 3.3.6, forced decoding is used to force the SMT decoder to produce only the reference correction using all possible phrase alignments in the phrase table. This can help us find out whether the SMT system can correct the errors using the current phrase table despite decoding. A system using forced decoding will correct decoding errors, but not OOV errors. Results of forced decoding experiments by error type are also presented in Table 3.11.

Decoding errors: Contribute to about 54% of all missed errors. We observe that these errors often involve changes of only one or a few words (especially function words), where long-range contextual information may be needed for accurate correction. For example, 83% of missed *ArtOrDet* errors and 88% of missed *SVA* errors are corrected during forced decoding. Since SMT was not originally designed for GEC and many standard features may not perform well on this task, the highest-ranked candidate selected by the decoder is not always the best correction. Adding new local and global features to help the decoder distinguish good from bad corrections may overcome this problem. For example, the character-level Levenshtein distance feature has proved to be useful and effective in Section 3.4.1. Additionally, in our final hybrid system submitted to the CoNLL-2014 shared task, a large-scale LM is used to re-rank the 10-best candidates from the SMT system to help minimise SMT

[6]Noun, singular or mass → Noun, plural

decoding errors (see Section 3.5). Candidate re-ranking will be investigated in detail in Chapter 4.

OOV errors: Account for the remaining 46% missed errors. Useful phrase mappings required to correct these errors could not be learnt from the training data due to data sparsity. Our analysis reveals that these errors often involve rare words (erroneous or not), open-class words (e.g. nouns, verbs) or longer phrases. It is worth noting that mappings for phrases longer than 7 tokens are not able to be learnt or extracted by the system because the maximum phrase length is set to 7 during training. Apart from adding more training data (as discussed in Section 3.3.4), another solution is to use more generalised models. Neural network models are appealing for GEC as they may be able to correct errors in previously unseen phrases and sentences more effectively. We will address this in detail in Chapter 5.

3.5 Results in the CoNLL-2014 shared task

The CoNLL-2014 shared task on grammatical error correction required participating systems to correct all the errors present in text written by learners of English. Our submission used a hybrid approach, which includes:

- a rule-based system from the self-assessment and tutoring system developed at the University of Cambridge for helping intermediate learners of English in their writing tasks (Andersen et al., 2013) - RBS;[7]
- the final SMT system from Section 3.4;
- a large-scale Microsoft n-gram LM built from web documents (Gao et al., 2010) to rank alternative corrections;
- an error type filtering technique to filter out some unnecessary corrections.

Results for the individual systems (i.e. RBS and SMT) and different combinations of them on the development set are reported in Table 3.12. We can see that our SMT system (#2) has much better performance than the rule-based system (#1). Using the rule-based system as the first processing step to perform an initial correction helps the SMT system (#3), suggesting that some corrections from the rule-based system and SMT system are complementary. Performance is improved when the candidates generated from the rule-based system output are ranked by the LM before applying the SMT system (#4). As we have observed in Section 3.4.2.3, the candidate with the highest probability from the SMT system is not always the best correction. Using the LM to re-rank the 10-best candidates from the SMT system yields better performance (#5). Therefore, candidate re-ranking for SMT-based GEC systems seems necessary. Filtering out types with zero P (i.e. *Reordering*, *Srun* and *Wa* - see Table 3.11) improves overall P while preserving R (#6). Our

[7]The latest version of the system, called *Write & Improve*, is available at `https://sat.ilexir.co.uk`.

#	System	CE	ME	UE	P (%)	R (%)	$F_{0.5}$ (%)
1	RBS	95	3,322	107	47.03	2.78	11.24
2	SMT	452	2,965	690	39.58	13.23	28.30
3	RBS>SMT	476	2,941	738	39.21	13.93	28.77
4	RBS_c>LM>SMT	471	2,946	781	39.61	13.78	28.81
5	RBS_c>LM>$SMT_{10\text{-best}}$>LM	681	2,736	1366	33.27	19.93	29.34
6	RBS_c>LM>$SMT_{10\text{-best}}$>LM>Filter	681	2,736	1350	33.53	19.93	29.50

Table 3.12: Results for different systems on the CoNLL-2014 development set. Subscript 'c' indicates candidates generated from a system's individual corrections, subscript '10-best' indicates the 10-best list of candidates produced by the SMT system, and '>' indicates a pipeline where the output of one system is the input to the other.

submission to the CoNLL-2014 shared task is the result of our best hybrid system, that is RBS_c>LM>$SMT_{10\text{-best}}$>LM>$Filter$ (#6).

The official test set comprises 50 new essays (approximately 30,144 tokens in 1,312 sentences) written in response to two prompts, one of which was also included in the training data. Two official rounds of evaluation were performed. The first was based on the original gold-standard annotations made by two human annotators independently, whereas the second was based on a revised version that includes alternative annotations submitted by the participating teams. The official results for the 13 submissions in both evaluation rounds are reported in Table 3.13. Our submitted system (*CAMB*) achieved first and second place respectively.

A closer observation of the system's output and the gold-standard annotation reveals a number of cases where the system introduces changes that are not part of the gold standard but we consider improve the quality of a sentence, suggesting that true system performance is underestimated. For example:

Example 3.19. Uncredited correction:

ORIGINAL SENTENCE ***Demon*** *is not easily* ***to be*** *defeated and it* ***is required much of*** *energy and psychological support.*

SYSTEM OUTPUT ***Demon*** *is not easily defeated and it* ***requires a lot of*** *energy and psychological support.*

GOLD STANDARD ***The demon*** *is not easily defeated and it* ***requires much*** *energy and psychological support.*

Adding alternative corrections to the gold standard alleviates this problem, although the list of alternatives will inevitably be incomplete.

There are also a number of cases where the sentences are considered incorrect as part of a longer text but are acceptable when they are evaluated in isolation. Consider the following example:

Example 3.20. Uncredited correction:

ORIGINAL SENTENCE ***It has*** *erased the boundaries of distance and time.*

SYSTEM OUTPUT ***It has*** *erased the boundaries of distance and time.*

GOLD STANDARD ***They have*** *erased the boundaries of distance and time.*

Team	Original test set				Revised test set			
	P (%)	**R** (%)	**$F_{0.5}$** (%)	**Rank**	**P** (%)	**R** (%)	**$F_{0.5}$** (%)	**Rank**
CAMB	39.71	30.10	37.33	1	46.70	34.30	43.55	2
CUUI	41.78	24.88	36.79	2	52.44	29.89	45.57	1
AMU	41.62	21.40	35.01	3	45.68	23.78	38.58	3
POST	34.51	21.73	33.88	4	41.28	25.59	36.77	4
NTHU	35.08	18.85	29.92	5	38.34	21.12	32.97	6
RAC	33.14	14.99	26.68	6	35.63	16.73	29.06	8
UMC	31.27	14.46	25.37	7	43.17	19.72	34.88	5
PKU*	32.21	13.65	25.32	8	36.64	15.96	29.10	7
NARA	21.57	29.38	22.78	9	23.83	31.95	25.11	9
SJTU	30.11	5.10	15.19	10	32.95	5.95	17.28	10
UFC*	70.00	1.72	7.84	11	72.00	1.90	8.60	11
IPN*	11.28	2.85	7.09	12	11.66	3.17	7.59	12
IITB*	30.77	1.39	5.90	13	34.07	1.66	6.94	13

Table 3.13: CoNLL-2014 official evaluation results (Ng et al., 2014). The teams that submitted their system output after the deadline have an asterisk (*) after their team names.

The system candidate is perfectly grammatical on its own (at the sentence level), but it is considered incorrect when analysed in context (at the script level). Such mismatch is the result of discrepancies between the annotation and evaluation criteria: while the gold standard is annotated taking discourse into account, system corrections are proposed in isolation, completely devoid of discursive context.

The shared task results confirm our hypothesis that the SMT approach is suitable for an all-errors correction task. Our SMT system forms the basis of a state-of-the-art all-errors GEC system. Another important thing to note is that despite the low $F_{0.5}$ scores in Table 3.13, Bryant and Ng (2015) reported that our system, *CAMB*, was able to perform 73% as reliably as a human annotator when further alternative corrections are taken into account.

3.6 Summary

In this chapter, we have investigated SMT for GEC. We have shown that SMT can form the basis of a competitive all-errors GEC system. We have explored different TMs, LMs and alignment methods used in the SMT system. To overcome the lack of training data, we have investigated three alternative sources of data: 1) parallel sentences extracted from other high-quality learner corpora; 2) artificial data generated by injecting errors into error-free English sentences; and 3) short parallel phrases extracted from error annotations. In addition, we have demonstrated that phrase table filtering can be used to improve system performance. A phrase-based SMT system has proved to be effective, and it forms one half of our winning system submitted to the CoNLL-2014 shared task.

In order to better understand the performance of the SMT-based GEC system, we have performed a detailed error analysis. The SMT system is particularly good at correcting errors that have more training examples, involve changes of only one or a few words and depend on local context. In terms of error types, the system achieves the best performance for *Wform*, *Mec*, *Nn* and *ArtOrDet*. We have also shown that the SMT system can correct sequential errors and interacting errors at the same time. However, about 54% of missed errors are caused by decoding errors while the remaining 46% are missed due to OOV errors.

chapter 4

Candidate re-ranking

We observe that many errors are missed by the system developed in the previous chapter. Better corrections are in the n-best list of candidates produced by the system, but the decoder fails to select the best correction. In this chapter, we develop a supervised ranking model to re-rank candidates generated by an SMT-based GEC system. A range of novel features with respect to GEC are investigated and implemented in our re-ranker. We train a rank preference SVM model and demonstrate that this outperforms two other ranking models for GEC. Experimental results on the publicly available FCE dataset show that our re-ranker can help improve sentence quality.

The work presented in this chapter was published in the 11th Workshop on Innovative Use of NLP for Building Educational Applications, North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Yuan et al., 2016).

4.1 Introduction

As demonstrated in the previous chapter, SMT can form the basis of a competitive all-errors GEC system. However, the best candidate produced by an SMT-based GEC system is not always the best correction, as illustrated in the following examples:[1]

Example 4.1. *There <NS type="AGV">are<c>is</c></NS> some <NS type="CN">informations<c>information</c></NS> you have asked me about.*

Here, 'are' should be changed to 'is' (*"AGV"* stands for Verb AGreement) and 'informations' should be corrected to 'information' (*"CN"* stands for Countability of Noun).

[1]Sentences are taken from the FCE dataset and annotated using the CLC error-coding scheme.

Original sentence	*There **are** some **informations** you have asked me about.*
Smt output	
1st	*There are some information you have asked me about.*
2nd	*There is some information you have asked me about.*
3rd	*There are some information you asked me about.*
4th	*There are some information you have asked me.*
5th	*There are some information you have asked me for.*
6th	*There are some information you have asked me about it.*
7th	*There is some information you asked me about.*
8th	*There are some information you asked me for.*
9th	*There were some information you have asked me about.*
10th	*There is some information you have asked me.*
Gold standard	*There **is** some **information** you have asked me about.*

The SMT output is the one with the highest probability (*1st*), which only corrects the mass noun error (*informations* → *information*), but misses the agreement error (*are* → *is*). However, the *2nd*-ranked candidate corrects both errors and matches the reference.

Example 4.2. *There will be signs to follow from <NS type="MD"><c>the</c></NS> <NS type="RP">Central<c>central</c></NS> train station.*

In this case, 'the' should be added (*"MD"* stands for Missing Determiner) and 'Central' should be changed to 'central' (*"RP"* stands for Replace Punctuation).

Original sentence	*There will be signs to follow from **Central** train station.*
Smt output	
1st	*There will be signs to follow from central train station.*
2nd	*There will be signs to follow from Central train station.*
3rd	*There will be signs to follow from the central train station.*
4th	*There will be signs to follow from Central the train station.*
5th	*There will be signs to follow from the Central train station.*
6th	*There will be signs to follow , from central train station.*
7th	*There will be signs to follow , from Central train station.*
8th	*There will be signs to follow from the Central the train station.*
9th	*There will be signs to follow from central the train station.*
10th	*There will be a signs to follow from central train station.*
Gold standard	*There will be signs to follow from **the central** train station.*

The *3rd*-ranked candidate is better than the SMT output (*1st*) as it not only corrects the capitalisation error (*Central* → *central*) but also inserts the determiner 'the'.

Since SMT was not originally designed for GEC, many standard features do not perform well on this task. Thus, it is necessary to add new local and global features to help the decoder distinguish good from bad corrections. We used Levenshtein distance to limit the changes made by our SMT system, given that most words translate into themselves and errors are often similar to their correct forms (see Section 3.3.5 and 3.4.1). Junczys-Dowmunt and Grundkiewicz (2014) also augmented their SMT system with the word-level Levenshtein distance features.

However, the integration of additional models/features into the decoding process may affect the dynamic programming algorithm used in SMT, since it does not support such complex features as those computed from an n-best list. An alternative

to performing this 'integrated decoding' is to re-rank the translation candidates produced by the SMT system using a rich set of features that are not used by the SMT decoder, so that better candidates can be selected as 'optimal' translations. This has several advantages: 1) it allows the introduction of new features that are tailored for GEC; 2) unlike in SMT, we can use various types of features without worrying about fine-grained *smoothing* issues and it is easier to use global features; 3) re-ranking is easy to implement and the existing decoder does not need to be modified; and 4) the decoding process in SMT only needs to be performed once, which allows for fast experimentation.

Most previous work on GEC has used evaluation methods based on P, R or F-score, as in the latest shared tasks. However, as discussed in Section 2.4, these evaluation methods do not provide an indicator of improvement on the original text so there is no way to compare GEC systems with a 'do-nothing' baseline. Since the aim of GEC is to improve text quality, we use the I-measure, which tells us whether a system improves the input.

The main contributions of our work are as follows. First, to the best of our knowledge, we are the first to use a supervised discriminative re-ranking model in SMT for GEC, showing that n-best list re-ranking can be used to improve sentence quality. Second, we propose and investigate a range of easily computed features for GEC re-ranking. Finally, we report results on several well-known publicly available test sets that can be used for cross-system comparisons.

4.2 Approach

Our re-ranking approach is defined as follows:

1. an SMT system is first used to generate an n-best list of candidates for each input sentence;
2. features that are potentially useful to discriminate between good and bad corrections are extracted from the n-best list;
3. these features are then used to determine a new ranking for the n-best list;
4. the new highest-ranked candidate is finally output.

The SMT system is not perfect, so candidates with the highest probability do not always constitute the best correction. For this reason, an n-best list re-ranker is trained to find better corrections. We treat n-best list re-ranking as a discriminative ranking problem. Unlike in standard SMT, the source input sentence is also added to the candidate pool if it is not in the n-best list, since in many cases the source sentence has no errors and should be translated as itself.

We use rank preference SVMs (Joachims, 2002) in the SVM^{rank} package (Joachims, 2006), an efficient implementation of the SVM framework (Vapnik, 1995). This model learns a ranking function from preference training examples and then assigns a score to each test example, from which a global ordering is derived. The default linear kernel is used due to training and testing time costs.

SVMs are widely used for learning classification, regression, or ranking functions. The basic idea of SVMs is to find a maximum (soft-)margin hyperplane that can separate two different classes correctly, and simultaneously maximise the (soft-)margin (or the distance) between that hyperplane and other 'difficult points' close to it. These 'difficult points' are called support vectors, and a decision function is fully specified by these support vectors. Given a set of instance-label pairs (x_i, y_i), where $x_i \in R^n$, $y_i \in \{1, -1\}$ for $i = 1, ..., l$, any hyperplane can be written as the set of points x satisfying

$$w^T x + b = 0 \tag{4.1}$$

where w is the normal vector to the hyperplane (known as the weight vector) and b is the bias. Learning the SVM can then be formulated as a constrained optimisation problem:

$$\min_{w,b} \ \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w, b; x_i, y_i) \tag{4.2}$$

subject to

$$y_i(w^T x_i + b) \geq 1 - \xi(w, b; x_i, y_i) \text{ for } i = 1, ..., l \tag{4.3}$$

where a penalty parameter C allows a trade-off between the margin size and the training error, and slack variables $\xi(w, b; x_i, y_i)$ measure the extent of misclassification. As $\xi(w, b; x_i, y_i) \geq 0$, the constraint in Equation 4.3 is equivalent to

$$\xi(w, b; x_i, y_i) = \max(0, 1 - y_i(w^T x_i + b)) \tag{4.4}$$

Therefore, the learning problem is equivalent to the unconstrained optimisation problem:

$$\min_{w,b} \ \frac{1}{2} w^T w + C \sum_{i=1}^{l} \max(0, 1 - y_i(w^T x_i + b)) \tag{4.5}$$

Rank preference SVMs work as follows. Suppose that we are given a set of ranked instances R containing training samples x_i and their target rankings r_i:

$$R = \{(x_1, r_1), (x_2, r_2), ..., (x_l, r_l)\} \tag{4.6}$$

such that $x_i \succ x_j$ when $r_i < r_j$, where $\succ$ denotes a preference relationship. A group of ranking functions are defined, where each function f determines the preference relations between instances:

$$x_i \succ x_j \Leftrightarrow f(x_i) > f(x_j) \tag{4.7}$$

The aim is to find the best function f that minimises a given loss function ξ with respect to the given ranked instances. Instead of using the R set directly, a set of pairwise difference vectors are created and used to train a model. For linear ranking models, this is equivalent to finding the weight vector w that maximises the number

of correctly ranked pairs:

$$\forall(x_i \succ x_j) : w(x_i - x_j) > 0 \tag{4.8}$$

which is, in turn, equivalent to solving the following optimisation problem:

$$\min_w \ \frac{1}{2} w^T w + C \sum \xi_{ij} \tag{4.9}$$

subject to

$$\forall(x_i \succ x_j) : w(x_i - x_j) \geq 1 - \xi_{ij} \tag{4.10}$$

where $\xi_{ij} \geq 0$.

4.3 Feature space

New features are introduced to identify better corrections in the n-best list produced by the SMT decoder. These are described briefly below.

4.3.1 SMT feature set

The SMT feature set reuses information extracted from the SMT system. As the SMT framework has been shown to produce good results for GEC, we reuse these pre-defined SMT features. This feature set includes:

4.3.1.1 Decoder's scores

The SMT decoder's scores include unweighted TM scores, reordering model scores, LM scores and word penalty scores. We use unweighted scores, as their weights will be reassigned during training.

4.3.1.2 N-best list ranking information

The n-best list feature set encodes the original ranking information provided by the SMT decoder. Both *linear* and *non-linear* transformations are used.

Note that both the decoder's features and the n-best list ranking features are extracted from the SMT system output. If the source sentence is not in the n-best list, it will not have these two kinds of features and zeros will be used.

4.3.2 Language model feature set

Raw candidates from an SMT system can include many malformed sentences so we introduce LM features and adaptive language model (ALM) features in an attempt to identify and discard them.

4.3.2.1 LM features

LMs are widely used in GEC, especially to rank correction suggestions proposed by other models. Ideally, correct word sequences will get high probabilities, while incorrect or unseen ones will get low probabilities. We use Microsoft's Web N-gram Services, which provide access to large *smoothed* n-gram LMs built from web documents. All our experiments are based on the 5-gram 'bing-body:apr10' model, the same one used in Section 3.5. We also build several n-gram LMs (for $n = 3$, 4, and 5) from native and learner corpora, including the corrected version of the CLC, the written part of the BNC and ukWaC (Ferraresi et al., 2008). UkWaC is a very large corpus of English (with more than 2 billion tokens) constructed by crawling the `.uk` Internet domain. The LM feature set contains unnormalised sentence scores, normalised scores using *arithmetic mean* and *geometric mean*, and the minimum and maximum n-gram probability scores.

4.3.2.2 ALM features

ALM scores are calculated from the n-best list's n-gram probabilities. N-gram counts are collected using the entries in the n-best list for each source sentence. N-grams repeated more often than others in the n-best list get higher scores, thus ameliorating incorrect lexical choices and word order. The n-gram probability for a target word c_i given its history c_{i-n+1}^{i-1} is defined as:

$$P_{n-best}(c_i|c_{i-n+1}^{i-1}) = \frac{count_{n-best}(c_{i-n+1}^{i})}{count_{n-best}(c_{i-n+1}^{i-1})} = \frac{count_{n-best}(c_{i-n+1}^{i})}{\sum_{c_i} count_{n-best}(c_{i-n+1}^{i})} \tag{4.11}$$

For a sentence C with l words:

$$P_{ALM}(C) = \log(\prod_{i=1}^{l} P_{n-best}(c_i|c_{i-n+1}^{i-1})) \tag{4.12}$$

We then normalise the score by the sentence length l to get an average word log probability, making it comparable for sentences of different lengths. In our re-ranking system, different values of n are used, from 2 to 6. This feature is taken from Hildebrand and Vogel (2008).

4.3.3 Statistical word lexicon feature set

We use the word lexicon learnt by IBM Model 4 (see Section 3.1.2.1), which contains translation probabilities for word-to-word mappings. The statistical word translation lexicon is used to calculate the translation probability $P_{lex}(c_i)$ for each word c_i in the target sentence C. $P_{lex}(c_i)$ is the sum of all translation probabilities of c_i for each word e_j in the source sentence E. Specifically, this can be defined as:

$$P_{lex}(c_i|E) = \frac{1}{m+1} \sum_{j=0}^{m} P(c_i|e_j) \tag{4.13}$$

where m is the source sentence length. $P(c_i|e_j)$ is the word-to-word translation probability of the target word c_i from one source word e_j.

As noted by Ueffing and Ney (2007), the sum in Equation 4.13 is dominated by the maximum lexicon probability, which we also use as an additional feature:

$$P_{lex-max}(c_i|E) = \max_{j=0,\ldots,m} P(c_i|e_j) \tag{4.14}$$

For both lexicon scores, we sum over all words c_i in the target sentence and normalise by sentence length to get sentence translation scores. Lexicon scores are calculated in both directions. This feature is also taken from Hildebrand and Vogel (2008).

4.3.4 Levenshtein distance feature set

A close observation reveals that raw candidates from an SMT system can also include fluent sentences that change the source significantly. In GEC, we want to keep the original sentences written by learners as much as possible, and make only the minimum number of necessary corrections. Therefore, we may want to limit the changes made by the system. As discussed in Section 3.3.5, Levenshtein distance is a string metric for measuring the difference between two sequences, reflecting the minimum number of edits (i.e. insertions, deletions or substitutions) required to change one sequence into another. Both word-level and character-level similarity scores are calculated using Equation 3.34. The overall scores and breakdowns are used as features. It is worth noting that character-level Levenshtein distance is also used as a feature in our SMT system.

4.3.5 Length feature set

These features are used to make sure that the final system does not make unnecessary deletions or insertions. This set contains four length ratios:

$$score(H_s, E) = \frac{N(H_s)}{N(E)} \tag{4.15}$$

$$score(H_s, H_1) = \frac{N(H_s)}{N(H_1)} \tag{4.16}$$

$$score(H_s, H_{max}) = \frac{N(H_s)}{N(H_{max})} \tag{4.17}$$

$$score(H_s, H_{min}) = \frac{N(H_s)}{N(H_{min})} \tag{4.18}$$

where H_s is the sth candidate, E is the source (erroneous) sentence, H_1 is the 1-best candidate (the candidate ranked *1st* by the SMT system), $N(\cdot)$ is the sentence's length, $N(H_{max})$ is the maximum candidate length in the n-best list for that source sentence and $N(H_{min})$ is the minimum candidate length.

4.3.6 Syntactic vs. non-syntactic

We decide to use only non-syntactic features for a number of reasons. Firstly, non-syntactic features are easier to compute, while syntactic features depend on syntactic analysis. In addition, non-syntactic features extracted for the candidates in the n-best list are more reliable than syntactic features (e.g. features based on parser output) as most existing NLP tools do not perform well with sentences containing errors (see Section 3.2). Last but not least, previous work has shown that non-syntactic features seem more effective than syntactic features when re-ranking SMT n-best lists (see Section 2.2.2).

4.4 Experiments

4.4.1 Experimental set-up

We use the publicly available FCE dataset, which is a part of the CLC. As discussed in Section 2.3.2.1, the FCE dataset is a set of 1,244 scripts written by learners of English who took the FCE examination between 2000 and 2001. The texts have been manually error-annotated with a taxonomy of approximately 80 error types (Nicholls, 2003). As discussed in Section 2.6, the reasons for using the FCE dataset instead of the NUCLE corpus are: 1) the FCE dataset is a more representative test set of learner writing as it covers a wide variety of L1s; 2) the error annotations in the NUCLE corpus are sometimes unreliable and inconsistent; 3) the FCE dataset was annotated using the same annotation scheme as the CLC; and 4) results reported on the publicly available FCE dataset can be used for cross-system comparisons.

Following Yannakoudakis et al. (2011), we split the publicly available FCE dataset into training and test sets: we use the 1,147 scripts from the year 2000 for training and the 97 scripts from the year 2001 for testing. The sizes of the FCE training and test sets are given in Table 4.1. Both the FCE and NUCLE training sets are too small to build good SMT systems, considering that previous work has shown that training on small datasets does not work well for SMT-based GEC and adding more training data helps (e.g. see Section 3.3.4; Yuan and Felice (2013); Felice et al. (2014); Junczys-Dowmunt and Grundkiewicz (2014)). To overcome this problem, we use examples extracted from the fully error-coded CLC (approximately 1,934,732 pairs of parallel sentences and 28,722,561 tokens on the target side).

Segmentation and tokenisation are performed using the Robust Accurate Statistical Parsing (RASP) system (Briscoe et al., 2006), which is expected to perform better in the noisy domain of learner text than systems developed from high quality copy-edited text.

System performance is evaluated in terms of I-measure, which is designed to address problems with previous evaluation methods and reflect any improvement on the original sentence after applying a system's corrections.

Dataset	Scripts	Sentences	Tokens
Training set	1,147	30,995	496,567
Test set	97	2,691	41,986

Table 4.1: The FCE dataset sizes.

4.4.2 SMT system

We train several new SMT systems based on the FCE dataset and select the best one for our re-ranking experiments. These systems use different configurations and improved methods proposed in the previous chapter, defined as follows:

- alignment methods:
  - GIZA++:
    uses GIZA++ for word alignment;
  - Pialign:
    uses Pialign to learn a phrase table;
  - our method:
    uses our method to build a phrase table directly - see Section 3.3.5;
- training data:[2]
  - FCE:
    uses the publicly available FCE dataset as training data;
  - CLC:
    incorporates sentence-level training data extracted from the CLC;
  - phrase:
    uses short phrase alignments extracted from error annotations, where identical phrase pairs are removed - see Section 3.3.4.3;
- LD:

  limits edit distance by adding character-level Levenshtein distance as a new feature;
- LM:

  builds a bigger LM by adding the corrected version of the CLC for decoding.

[2]The artificial datasets used in Section 3.3.4.2 are not used here as they are generated for the NUCLE data (i.e. the error patterns and error distributions used to inject errors into error-free text are learnt from the NUCLE corpus) and so are not expected to perform well on the FCE data.

System performance is shown in Table 4.2. We also report results using other evaluation metrics for comparisons: $F_{0.5}$ and F_1 from the M^2 scorer, GLEU and BLEU. *Baseline* is a baseline system which makes no corrections. As mentioned earlier in Section 2.4, it always gets a zero F-score. We see that not all the systems make the source text better (i.e. they do not have positive I scores). In addition, not all the improved methods that were useful in the previous chapter yield better I scores on the FCE test set. Pialign outperforms GIZA++ and our method. Adding more learner examples improves system performance (*+CLC*), as does a bigger LM for decoding (*+LM*) and the Levenshtein distance feature (*+LD*). However, adding short parallel phrases extracted from error annotations into the training set yields lower I scores (*+phrase*). As discussed in Section 3.3.4.4, these extracted short phrases are mainly used to boost the probability of phrase alignments that involve corrections, so as to improve R (at the cost of P, though). Adding these parallel phrases consistently yields better R and F_1. The best system in terms of I-measure is the one that uses the whole CLC, Pialign, a bigger in-domain LM for decoding, and edit distance as an additional feature (*Pialign+FCE+CLC+LM+LD*). The positive I score of 2.87% shows a real improvement in sentence quality. This system is also the best in terms of BLEU (80.52%), GLEU (70.15%) and $F_{0.5}$ (52.90%).[3] Therefore, we use the n-best list from this system to perform re-ranking.

4.4.3 SVM re-ranker

The input to the re-ranking model is the 10-best list output from an SMT system. The original source sentence is used to collect a 10-best list of candidates generated by the SMT decoder, which is then used to build a supervised re-ranking model.

4.4.3.1 Assigning gold labels

In order to train SVM re-rankers, we need a gold ranking of correction candidates for each source sentence. Since we do not have human judgements for n-best lists, we approximate two versions of the sentence-level rankings using WAcc and I scores as the ranking metric respectively. We then build two SVM re-rankers using all the features defined in Section 4.3:

- SVM_{WAcc}:

  uses sentence-level WAcc scores as gold labels;

- SVM_I:

  uses sentence-level I scores as gold labels.

Re-ranking results for the two SVM systems on the FCE test set are presented in Table 4.3. The effectiveness of our SVM re-rankers is evident, as performing a 10-best list re-ranking yields a substantial improvement in performance over the top-ranked output from our best SMT system from Section 4.4.2 (*SMT* in Table 4.3). Using sentence-level I scores (SVM_I) outperforms WAcc (SVM_{WAcc}). Therefore, in the experiments reported hereafter, we use sentence-level I scores as gold labels.

[3]The best system in terms of F_1 is *Pialign+FCE+CLC+LM+phrase*, with an F_1 score of 43.50%.

Alignment	Setting	BLEU	GLEU	M^2				I-measure	
				P	R	$F_{0.5}$	F_1	WAcc	I
Baseline		75.24	60.39	-	0	0	0	86.83	0
GIZA++	FCE	73.46	61.42	36.66	16.97	29.76	23.20	83.24	-4.14
	FCE+LM	74.89	64.16	43.48	24.37	37.58	31.23	83.99	-3.27
	FCE+LD	73.88	61.64	37.70	16.40	29.92	22.86	83.64	-3.68
	FCE+LM+LD	75.27	64.36	45.01	23.71	38.16	31.06	84.41	-2.79
	FCE+CLC+LM	76.47	67.70	48.67	37.64	45.97	42.45	83.94	-3.33
	FCE+CLC+LM+LD	**76.91**	**67.98**	**49.87**	37.16	**46.67**	42.59	**84.42**	**-2.78**
	FCE+phrase	70.54	59.30	30.76	19.84	27.71	24.12	80.64	-7.13
	FCE+LM+phrase	73.40	63.49	40.44	27.50	36.96	32.74	82.64	-4.82
	FCE+LD+phrase	71.44	59.93	32.09	19.16	28.27	23.99	81.41	-6.24
	FCE+LM+LD+phrase	74.07	63.92	41.87	26.63	37.57	32.55	83.31	-4.05
	FCE+CLC+LM+phrase	74.64	66.74	44.91	**40.53**	43.96	42.61	82.37	-5.13
	FCE+CLC+LM+LD+phrase	75.35	67.19	46.14	39.74	44.70	**42.70**	83.00	-4.42
Pialign	FCE	75.10	62.22	43.13	11.34	27.64	17.96	84.94	-2.17
	FCE+LM	75.69	63.40	45.19	15.24	32.44	22.79	83.10	-4.29
	FCE+LD	75.09	62.19	43.07	11.17	27.41	17.74	85.00	-2.11
	FCE+LM+LD	75.58	63.20	44.59	14.45	31.47	21.83	83.10	-4.30
	FCE+CLC+LM	80.39	70.07	62.37	32.19	52.52	42.46	87.01	1.38
	FCE+CLC+LM+LD	**80.52**	**70.15**	**63.27**	31.95	**52.90**	42.46	**87.21**	**2.87**
	FCE+phrase	72.04	60.38	31.89	17.36	27.32	22.48	80.66	-7.11
	FCE+LM+phrase	74.33	63.50	40.54	22.29	34.84	28.76	80.86	-6.88
	FCE+LD+phrase	72.38	60.63	32.26	16.93	27.31	22.21	80.99	-6.73
	FCE+LM+LD+phrase	74.46	63.57	40.92	22.01	34.92	28.62	81.03	-6.68
	FCE+CLC+LM+phrase	78.17	69.16	51.08	**37.88**	47.75	**43.50**	84.65	-2.52
	FCE+CLC+LM+LD+phrase	78.31	69.21	51.58	37.51	47.98	43.43	84.83	-2.30
Our method	FCE	69.70	58.59	29.49	18.83	26.49	22.98	80.54	-7.25
	FCE+LM	72.66	62.61	38.56	26.17	35.22	31.18	82.35	-5.16
	FCE+LD	71.54	59.91	31.78	17.36	27.26	22.45	82.02	-5.54
	FCE+LM+LD	73.56	63.15	40.26	24.72	35.77	30.63	**83.17**	**-4.22**
	FCE+CLC+LM	72.74	64.30	40.80	**36.90**	39.95	38.75	81.14	-6.55
	FCE+CLC+LM+LD	**74.15**	**65.37**	**43.01**	36.11	**41.43**	**39.26**	82.43	-5.07

Table 4.2: SMT system performance on the FCE test set (in percentages). The best results for each alignment method are marked in **bold**.

Model	WAcc	I
SMT	87.21	2.87
SVM_{WAcc}	87.90	8.10
SVM_{I}	**88.05**	**9.15**

Table 4.3: 10-best list re-ranking using different gold labels on the FCE test set (in percentages). The best results are marked in **bold**.

4.4.3.2 The feature set impact

In order to measure the contribution of each feature set to the overall improvement in sentence quality, a number of ablation tests are performed, where new models are built by removing one feature type at a time. If a decrease in performance is observed after removing a feature type, we then know that the feature type that has been removed has a positive effect on the overall performance. The bigger the difference in performance, the more important the feature type. However, if an increase in performance is observed, it suggests that the feature type has a negative effect and should not be used to build the re-ranking model.

#	**Feature**	**WAcc**	**I**
0	SMT	87.21	2.87
1	FullFeat	88.05	9.15
2	-SMT(decoder)	87.28	3.40
3	-SMT(rank)	87.82	7.47
4	-LM	87.93	8.33
5	-ALM	87.90	8.12
6	-word lexicon	87.75	6.98
7	-LD	**88.12**	**9.78**
8	-length	87.92	8.25
9	-LD-SMT(decoder)	87.25	3.20
10	-LD-SMT(rank)	87.93	8.32
11	-LD-LM	88.09	9.56
12	-LD-ALM	87.93	8.35
13	-LD-word lexicon	87.84	7.65
14	-LD-length	88.02	9.03
15	SMT(decoder)	87.15	2.40

Table 4.4: 10-best list re-ranking using different features on the FCE test set (in percentages). The best results are marked in **bold**.

In Table 4.4, *SMT* is the best SMT system output without re-ranking. *FullFeat* combines all feature types described in Section 4.3. The rest are *FullFeat* minus the indicated feature types. The first round of ablation tests (#2-8) tells us that not all the features in the *FullFeat* set have positive effects on the overall performance. A new model built using all but the Levenshtein distance features achieves an I score of 9.78% (#7), outperforming the one built using *FullFeat* (#1). This indicates that the Levenshtein distance features are detrimental and bring performance down. The removal of all the other types of features yields worse performance, suggesting that they all contribute to the overall improvement in sentence quality.

Therefore, we perform another round of ablation tests on the system without the Levenshtein distance features (*-LD* #7). Results (#9-14) confirm our finding that all the other feature types have positive effects on overall performance. Among them, the SMT decoder's scores are the most effective, as their absence is responsible for a 6.58 decrease in I score (#9). The removal of the word lexicon features also accounts for a 2.13 decrease (#13), followed by the SMT n-best list ranking information (1.46 #10), the ALM features (1.43 #12), the length features (0.75 #14) and the LM features (0.22 #11).

In order to test the performance of the SMT decoder's scores on their own, we built a new re-ranking model using only these features, which we report in Table 4.4 #15. We can see that using only the SMT decoder's scores as features yields worse performance than no re-ranking (#0), suggesting that the existing features used by the SMT decoder are not optimal when used outside the SMT ecosystem. We hypothesise that this might be caused by the lack of scores for the source sentences

Model	WAcc	I
SMT	87.21	2.87
SVM	88.12	9.78
Oracle	92.67	44.35

Table 4.5: Performance of SMT best, SVM re-ranker and oracle best (in percentages).

that are not included in the n-best list of the original SMT system. Therefore, the introduction of other types of features is necessary.

Overall, the best SVM re-ranker is built using all but the Levenshtein distance features, achieving an I score of 9.78% (#7).

4.4.4 Oracle score

In order to estimate a realistic upper bound on the task, we calculate an oracle score from the same 10-best list generated by our best SMT model. The oracle set is created by selecting the candidate which has the highest sentence-level WAcc for each source sentence in the test set.[4]

Table 4.5 compares the results of the SMT system (i.e. the best SMT model from Section 4.4.2), the SVM re-ranker (i.e. the best re-ranking model from Section 4.4.3) and the approximated oracle. We see that the oracle score is about 41 points higher than the standard SMT score in terms of I-measure, and about 5 points higher in terms of WAcc, confirming that there are alternative candidates in the 10-best list that are not chosen by the SMT model. Our re-ranker improves the I score from 2.87% to 9.78%, and the WAcc score from 87.21% to 88.12%, a substantial improvement over the best SMT model. However, there is still much room for improvement.

The oracle score tells us that, under the most favourable conditions, our re-ranking models could only improve the original text by 44.35% at most. This also reveals that, in many cases, the correct translation is not in the 10-best list. Therefore, it would be impossible to retrieve the correct translation even if the re-ranking model was perfect.

4.4.5 Benchmark results

We also compare our proposed re-ranking method with two other methods: Minimum Bayes-risk (MBR) re-ranking and Multi-Engine Machine Translation (MEMT) candidate combination.

4.4.5.1 MBR re-ranking

MBR was first proposed by Kumar and Byrne (2004) to minimise the expected loss of translation errors under loss functions that measure translation performance. A set of loss functions that incorporate different levels of linguistic information can be

[4]Since the I-measure is computed after maximising system WAcc at the sentence level, we use WAcc to select candidates that can be used to create the oracle set.

N	WAcc	I
1	87.21	2.87
10	87.32	3.71
100	87.31	3.63
200	87.31	3.62
1,000	87.34	3.83

Table 4.6: Results of MBR re-ranking on the FCE test set (in percentages).

defined. Instead of using the model's best output, the one that is most similar to the most likely translations is selected.

To translate a source sentence E into a target sentence C, given a loss function L and a true distribution P, the decision rule that minimises Bayes Risk is defined as:

$$\hat{C} = \underset{C' \in C_c}{\arg\min} \sum_{C \in C_e} L(C, C')P(C|E) \tag{4.19}$$

where C_c is the candidate space and C_e is the evidence space. Typically, the same n-best list is used as the candidate space C_c and the evidence space C_e. Therefore, Equation 4.19 can be modified to:

$$\hat{i} = \underset{i \in \{1,2,\ldots,n\}}{\arg\min} \sum_{j=1}^{n} L(C_j, C_i)P(C_j|E) \tag{4.20}$$

MBR re-ranking can then be considered selecting a *consensus* candidate, the least 'risky' candidate which is closest on average to all the likely candidates.

In our experiments, we use the same n-best list from our best SMT model as the candidate set and the evidence set. A loss function based on WAcc is used during MBR re-ranking. Results of using n-best lists with n ranging from 10 to 1,000 are reported in Table 4.6. In the table, $n = 1$ is the best SMT system output (i.e. SMT). We can see that performing MBR re-ranking yields better I scores than the SMT model without re-ranking, suggesting that candidates selected by the SMT system are not always the best corrections and MBR re-ranking can be used effectively to re-rank candidates for GEC. However, increasing the n-best list size does not produce a consistent improvement. Re-ranking the top 1,000 candidates yields the best performance, followed by 10, 100 and 200. As we use the same n-best list for the candidate space and the evidence space, we notice that some unreliable candidates are used as evidence (i.e. most likely translations) when we increase the n-best list size. In addition, using a bigger n-best list size enlarges the search space and therefore increases the searching time.

4.4.5.2 MEMT candidate combination

The MEMT system combination technique was first proposed by Heafield and Lavie (2010) and was successfully applied to GEC by Susanto et al. (2014). MEMT system combination is the process of combining the output of multiple systems to produce

a version that is better than each of its individual components. After combining the output of two classification-based GEC systems and two SMT-based GEC systems, Susanto et al. (2014) reported an $F_{0.5}$ score of 39.39% on the CoNLL-2014 shared task test set.

Following the work of Susanto et al. (2014), we decide to use the MEMT technique to combine the candidates in the n-best list from our best SMT system with the source sentence. During candidate combination, it is important to find the right alignments from the candidates, as alignment errors may result in ungrammatical sentences. We use METEOR (Banerjee and Lavie, 2005) to perform word alignment. Unlike GIZA++ and Pialign, METEOR aligns two sentences in the same language. The latest METEOR 1.5 only supports a few languages, and English is one of them. METEOR identifies not only words with exact matches, but also words with identical stems, synonyms, and unigram paraphrases. This is helpful for GEC as it can deal with word form, noun number, and verb form corrections that share identical stems, as well as word choice corrections with synonyms or unigram paraphrases. A *confusion network* is then constructed using the alignment information from METEOR. A *beam search* is later performed to find the best candidate. Features used by MEMT during *beam search* are:

- length:

  the candidate's length, which is used to normalise the impact of sentence length;

- LM:

  the LM score computed by the LM built from the corrected version of the CLC;

- back-off:

  the average n-gram length found in the LM;

- match:

  the number of n-gram matches between the compared sentences.

Results of combining the source sentence and the n-best candidates ($n = 1, 2, ...,$ 10) are presented in Table 4.7. *1-best* is the best SMT system output (i.e. *SMT*). Combining the source sentence and the best SMT output (*source+1-best*) yields an improvement in I score over *1-best* (from 2.87% to 3.25%), suggesting that the SMT system sometimes fails to distinguish good sentences that do not need any correction from erroneous ones, or correct parts from erroneous parts. Adding more candidates further improves performance although the improvement is not consistent (e.g. *source+2-best* outperforms *source+3-best*, *source+7-best* outperforms *source+8-best*). The best I score is achieved by combining the source and the 10-best candidates (*source+10-best*).

Candidates	WAcc	I
1-best	87.21	2.87
source+1-best	87.48	3.25
source+2-best	87.65	4.49
source+3-best	87.57	3.94
source+4-best	87.61	4.22
source+5-best	87.58	4.02
source+6-best	87.59	4.09
source+7-best	87.65	4.56
source+8-best	87.61	4.18
source+9-best	87.71	4.97
source+10-best	87.75	5.34

Table 4.7: Results of MEMT candidate combination on the FCE test set (in percentages).

Model	WAcc	I
SMT	87.21	2.87
SVM	**88.12**	**9.78**
MBR	87.32	3.71
MEMT	87.75	5.34

Table 4.8: Performance of SMT best, SVM re-ranker, MBR re-ranking and MEMT candidate combination (in percentages). The best results are marked in **bold**.

4.4.5.3 Results

In order to compare SVM re-ranking, MBR re-ranking and MEMT candidate combination for SMT-based GEC, the same 10-best list from our best SMT model is used for our re-ranking experiments and results are presented in Table 4.8. *SMT* is the best SMT model from Section 4.4.2, *SVM* is the best re-ranking model from Section 4.4.3 (which uses all but the Levenshtein distance features), *MBR* is the MBR re-ranking model from Section 4.4.5.1, and *MEMT* is the MEMT candidate combination model from Section 4.4.5.2. We observe that our supervised ranking model achieves the best I score, followed by MEMT candidate combination and MBR re-ranking. Our method clearly outperforms the other two methods, showing its effectiveness in re-ranking candidates for SMT-based GEC.

4.5 Analysis and discussion

Looking at the SVM re-ranker's output reveals that there are some learner errors which are missed by the SMT system but are captured by the re-ranker:[5]

[5]Example sentences are taken from the FCE dataset and annotated using the CLC error-coding scheme.

Example 4.3. Missed *RP* (Replace Punctuation) and *AGV* (Verb AGreement error):

Original sentence	*I meet a lot of people on **internet** and it really **interest** me.*
Smt output	*I meet a lot of people on **the internet** and it really **interest** me.*
Svm output	*I meet a lot of people on **the Internet** and it really **interests** me.*
Gold standard	*I meet a lot of people on **the Internet** and it really **interests** me.*

Example 4.4. Missed *RV* (Replace Verb):

Original sentence	*And they **effect** everyone's life directly or indirectly.*
Smt output	*And they **effect** everyone's life directly or indirectly.*
Svm output	*And they **affect** everyone's life directly or indirectly.*
Gold standard	*And they **affect** everyone's life directly or indirectly.*

Example 4.5. Missed *AGN* (Noun AGreement error):

Original sentence	*Of course I will give you some more **detail** about the student conference.*
Smt output	*Of course I will give you some more **detail** about the student conference.*
Svm output	*Of course I will give you some more **details** about the student conference.*
Gold standard	*Of course I will give you some more **details** about the student conference.*

4.5.1 Results on the CoNLL-2014 shared task development set

We have only tested our SVM re-ranker on the FCE dataset so far. In order to test how well it generalises, we apply our best SVM re-ranker to the CoNLL-2014 shared task development set (i.e. the CoNLL-2013 test set). We re-rank the 10-best correction candidates from the final SMT-based GEC system from Section 3.4.

System performance is evaluated in terms of I-measure. GLEU and $F_{0.5}$ scores are reported as well in Table 4.9 for future cross-system comparisons. The SMT system used in our winning system submitted to the CoNLL-2014 shared task yields a negative I score of -2.60%. However, this result is likely to be affected by the fact that the SMT system was optimised for $F_{0.5}$ during development (as explained in Chapter 3), as it was the official evaluation metric for the shared task. Similar results were reported by Felice and Briscoe (2015) where only one out of 13 participating systems (namely, *UFC* in Table 3.13) produced a positive I score. Our SVM re-ranker helps improve sentence quality as using it to re-rank the 10-best candidates from the SMT system yields an improvement in I score (from -2.60% to -1.65%). We also observe an increase in GLEU from 60.90% to 61.12%. The $F_{0.5}$ score for the SVM re-ranker is slightly lower than that for the SMT system. This

System	GLEU	$F_{0.5}$	I
Baseline	59.14	0	0
SMT	60.90	28.30	-2.60
SMT + SVM	61.12	27.86	-1.65

Table 4.9: Results of the baseline, the SMT system and the best SVM re-ranker on the CoNLL-2014 development set (in percentages).

is probably because the SVM re-ranker was optimised for WAcc (not $F_{0.5}$) during development (see Section 4.4). Our results show that our SVM re-ranker generalises well when trained on one dataset (i.e. CLC) and tested on a different one (i.e. NUCLE). However, the CLC training data was tokenised with RASP, whereas the NUCLE data was preprocessed using NLTK. We expect these results might be further improved by retokenising the CoNLL-2014 development set to be consistent with the tokenisation of the CLC.

We also study the performance of our SVM re-ranker by error type by computing P, R and F-score using our type estimation strategy described in Section 3.4.2. Results for the SMT system and the SVM re-ranker are reported in Table 4.10. We can see that the SVM re-ranker generally produces higher P but lower R. Moreover, it yields higher P than the SMT system for the following error types: *ArtOrDet*, *Mec*, *Nn*, *Npos*, *Others*, *Pform*, *Prep*, *Rloc-*, *SVA*, *Vform*, *Vm*, *Vt*, *Woinc* and *Wci* (both P and R increase). Better $F_{0.5}$ scores are observed for *Npos*, *Others*, *Pform*, *Vform*, *Vm*, *Woinc* and *Wci*. We observe that the SVM re-ranker helps the SMT decoder select better corrections as it can not only capture missed errors, but also avoid unnecessary changes made by the SMT system, as in the following examples:[6]

Example 4.6. *Wci*:

Original sentence	*Besides, the elderly with less salary also have to lower their **living standard**; both the consequences can cause a decrease in social happiness level and drive the whole society in **to** a less stable situation.*
Smt output	*Besides, the elderly with less salary also have to lower their **quality of life**; both the consequences can cause a decrease in social happiness level and drive the whole society in a less stable situation.*
Svm output	*Besides, the elderly with less salary also have to lower their **standard of living**; both the consequences can cause a decrease in social happiness level and drive the whole society in a less stable situation.*
Gold standard	*Besides, the elderly with less salary also have to lower their **standard of living**; both the consequences can cause a decrease in social happiness level and drive the whole society in a less stable situation.*

[6]Sentences are taken from the NUCLE corpus and annotated using the NUCLE error-coding scheme.

Example 4.7. Unnecessary change:

ORIGINAL SENTENCE	*How can it be guaranteed that our information will not be abused?*
SMT OUTPUT	*How can it be **assured** that our information will not be abused?*
SVM OUTPUT	*How can it be guaranteed that our information will not be abused?*
GOLD STANDARD	*How can it be guaranteed that our information will not be abused?*

4.5.2 Results on the CoNLL-2014 shared task test set

We also apply our best SVM re-ranker trained on the CLC to the CoNLL-2014 shared task test set. We re-rank the 10-best candidates from our winning system in the shared task (i.e. *CAMB* in Table 3.13). Our re-ranker is evaluated using GLEU, $F_{0.5}$ and I-measure on the original test set, which only includes the gold-standard annotations independently made by two annotators. In order to ensure a fairer and more reliable evaluation, alternative answers proposed by participating teams based on their system output are ignored.[7] Our proposed re-ranking model (*SVM*) is compared with five other systems: the baseline, the top three systems in the shared task and the MEMT system from Susanto et al. (2014), which combined the output of two classification-based and two SMT-based systems using MEMT, and achieved a better $F_{0.5}$ score of 39.39% - see Table 4.11. We see that our best SVM re-ranker outperforms the top three systems on all evaluation metrics. It also achieves a comparable $F_{0.5}$ score to the MEMT system from Susanto et al. (2014) although our SVM re-ranker is not trained on the NUCLE data or optimised for $F_{0.5}$. This result shows that our model generalises well to other datasets. As discussed in Section 4.5.1, we also expect these results might be further improved by retokenising the test data with RASP.

4.6 Recent work

Following the same line of research, Mizumoto and Matsumoto (2016) have recently proposed a similar discriminative re-ranking approach to re-score the 10-best candidates from an SMT system. Differences between our work and theirs are: 1) we focussed on non-syntactic features, while they mainly used syntactic features; 2) we applied rank preference SVMs, while they employed an averaged perceptron; and 3) our re-ranker was optimised for I-measure, while their system was tuned for $F_{0.5}$. The authors showed that re-ranking systems that use features extracted from POS and shallow parse tags improve performance and reported an $F_{0.5}$ score of 40.00% on the CoNLL-2014 shared task test set.

[7] Ng et al. (2014) observed that new scores tended to be biased towards the teams which submitted alternative answers.

Error type	SMT			SMT + SVM		
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
ArtOrDet	49.49	21.08	38.98	**51.15**	19.36	38.51
Cit	-	0.00	0.00	-	0.00	0.00
Mec	62.26	18.13	41.88	**66.67**	16.48	41.44
Nn	50.52	24.87	41.88	**51.72**	19.04	38.50
Npos	8.70	6.90	8.26	**18.18**	6.90	**13.70**
Others	25.00	5.71	14.93	**33.33**	5.71	**16.95**
Pform	9.09	3.85	7.14	**14.29**	3.85	**9.26**
Pref	33.33	2.56	9.80	0.00	-	0.00
Prep	38.16	9.35	23.62	**41.67**	8.06	22.73
Reordering	0.00	-	0.00	0.00	-	0.00
Rloc-	41.67	7.81	22.32	**46.15**	4.69	16.67
SVA	33.33	10.17	22.90	**36.00**	7.63	20.64
Sfrag	-	0.00	0.00	-	0.00	0.00
Smod	-	0.00	0.00	-	0.00	0.00
Spar	100.00	5.88	23.81	100.00	5.88	23.81
Srun	0.00	0.00	0.00	0.00	0.00	0.00
Ssub	33.33	4.23	14.02	25.00	2.82	9.71
Trans	21.74	3.55	10.73	20.00	2.13	7.46
Um	-	0.00	0.00	-	0.00	0.00
V0	25.00	5.56	14.71	20.00	5.56	13.16
Vform	32.76	16.10	27.14	**39.13**	15.25	**29.80**
Vm	50.00	8.42	25.16	**66.67**	8.42	**27.97**
Vt	37.14	8.39	22.03	**42.31**	7.10	21.24
Woadv	-	0.00	0.00	-	0.00	0.00
Woinc	3.85	5.41	4.08	**7.69**	5.41	**7.09**
Wa	0.00	0.00	0.00	0.00	0.00	0.00
Wci	13.27	3.05	8.52	**17.07**	**3.27**	**9.26**
Wform	58.62	22.57	44.43	57.32	20.80	42.42
Wtone	-	0.00	0.00	-	0.00	0.00
TOTAL	39.58	13.23	28.30	**43.44**	11.44	27.86

Table 4.10: Type-specific M^2 performance of the SMT system and the SVM re-ranker on the CoNLL-2014 development set (in percentages). Re-ranking improvements over the SMT system are marked in **bold**.

Instead of building a re-ranker, Hoang et al. (2016) trained a classifier to filter edits in the n-best list of candidates generated by an SMT system. Similar n-best list ranking information from the original SMT system, LM and POS features were used to train an edit classifier, which was later used to classify edits as valid or invalid. Final corrections were generated by discarding all invalid edits. Their method achieved an $F_{0.5}$ score of 41.19% by selecting edits from the 5-best candidates.

System	GLEU	$F_{0.5}$	I
Baseline	64.19	0	0
CAMB + SVM	65.68	38.08	-1.71
MEMT (Susanto et al., 2014)	n/a	39.39	n/a
Top 3 systems in CoNLL-2014			
CAMB (Felice et al., 2014)	64.32	37.33	-5.58
CUUI (Rozovskaya et al., 2014a)	64.64	36.79	-3.91
AMU (Junczys-Dowmunt and Grundkiewicz, 2014)	64.56	35.01	-3.31

Table 4.11: System performance on the CoNLL-2014 test set without alternative answers (in percentages).

4.7 Summary

In this chapter, we have investigated n-best list re-ranking for SMT-based GEC. We have shown that n-best list re-ranking can improve correction quality. A supervised machine learning model has been developed and shown to be effective and generalise well. We have defined a range of novel features with respect to GEC and systematically compared the contribution of different feature types to GEC re-ranking. We have trained a rank preference SVM model and demonstrated that it outperforms both MBR and MEMT based re-ranking for GEC. Our best re-ranking model achieves an I score of 9.78% on the publicly available FCE test set, compared to an I score of 2.87% for our best SMT system without re-ranking. The oracle score (upper bound) for re-ranking the 10-best list achieves over 40% I-measure performance, suggesting that further improvements may be possible. When testing on the official CoNLL-2014 shared task test set without alternative answers, our model achieves an $F_{0.5}$ score of 38.08%, an I score of -1.71%, and a GLEU score of 65.68%, outperforming the top three teams on all metrics.

CHAPTER 5

Neural machine translation for GEC

Our SMT-based GEC system suffers from data sparsity and many errors are not captured because useful phrase mappings could not be learnt from the training data (i.e. OOV errors). The recent success of neural network models provides the motivation for using NMT for GEC. In this chapter, we present the first tentative study using NMT for GEC. A two-step approach is proposed to handle the rare word problem in NMT, which has been proved to be useful and effective for GEC. Experimental results on publicly available datasets show that our NMT-based system can outperform our best SMT-based system.

Parts of the results presented in this chapter were published in the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Yuan and Briscoe, 2016).

5.1 Introduction

As discussed in Chapter 3, our phrase-based SMT system is trained on surface forms and makes little or no direct use of syntactic information. In order to make a correction, the exact phrase-level correction mapping (i.e. lexical pair) has to be seen in the training data. Consider the following example:[1]

Example 5.1. Missed *FN* (wrong Noun Form):

Original sentence	*... the automotive business:* ***tyres*** *recycling ...*
Smt output	*... the automotive business:* ***tyres*** *recycling ...*
Gold standard	*... the automotive business:* ***tyre*** *recycling ...*

Our SMT system is unaware of the underlying syntactic structure and cannot use the correction mapping *NNS* → *NN*.[2] Since the exact lexical pair *tyres* → *tyre*

[1] This and the following examples are taken from the FCE dataset and annotated using the CLC error-coding scheme.

[2] Noun, plural → Noun, singular or mass

is not in the phrase translation table learnt from the training data, our SMT system fails to correct the *FN* error.

NMT, as a recently proposed approach to MT, has shown promising results (see Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). Compared with conventional SMT, NMT has several advantages. First, unlike SMT which consists of components that are trained separately and combined during decoding (see Section 3.1), NMT learns a single large neural network which inputs a source sentence and outputs a translation. As a result, training NMT systems for end-to-end translation tasks is much simpler than building SMT systems, which requires multiple processing steps. Second, an NMT system can learn translation regularities directly from the training data, without the need to explicitly design features to capture them, which is quite difficult in SMT. Last but not least, the use of distributed representations for words in NMT helps alleviate the *curse of dimensionality*[3] by grouping similar words.

NMT is therefore appealing for the error correction task. When building GEC systems using the NMT framework, we no longer need to design new features with respect to GEC. In addition, NMT-based systems are able to correct unseen erroneous phrases and sentences more effectively. The individual erroneous words and their correct forms still need to be seen in the training data somewhere, but they do not need to be paired. For errors whose correction mappings could not be learnt from the training data (e.g. SMT OOV errors), NMT systems may have a chance to correct them if all the words involved are in the training data, as NMT does not rely on any correction mappings. NMT-based systems may thus help ameliorate the lack of large error-annotated learner corpora for GEC. However, like SMT-based systems, NMT-based systems are not capable of correcting errors involving rare words that have not been seen in the training data. For example:

Example 5.2. Missed *S* (Spelling):

ORIGINAL SENTENCE *... some remains from the Estruscans'* ***neoropolis****.*
GOLD STANDARD *... some remains from the Estruscans'* ***necropolis****.*

As the word 'necropolis' has not been seen in the training data, we believe that neither SMT-based systems nor NMT-based systems can correct this error.

One of the limitations of NMT is that systems typically limit vocabulary size on both source and target sides due to the complexity of training (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015b; Jean et al., 2015a). Therefore, they are unable to translate rare words, and unknown words are replaced with the *UNK* symbol. This problem is more serious for GEC, as non-native text contains not only rare words (e.g. proper nouns) but also misspelt words (i.e. spelling errors). By replacing all the unknown words with the same *UNK* symbol, useful information is discarded, resulting in systems that are not able to correct misspelt words or even keep some of the error-free original words. This is shown in the following examples, where words unknown to the NMT-based GEC system are underlined:

[3]A word sequence on which the model will be tested is likely to be different from all the word sequences seen during training (Bengio et al., 2003).

Example 5.3. Missed *S* (Spelling):

ORIGINAL SENTENCE	*... I am **goign** to make a plan ...*
NMT OUTPUT	*... I am [UNK] to make a plan ...*
GOLD STANDARD	*... I am **going** to make a plan ...*

Example 5.4. Unnecessary changes:

ORIGINAL SENTENCE	*I suggest you visit first the cathedral of "Le Seu d'Mrgell" because it is the most emblematic building in the area.*
NMT OUTPUT	*I suggest you visit first the cathedral of "Le [UNK] [UNK]" because it is the most [UNK] building in the area.*
GOLD STANDARD	*I suggest you visit first the cathedral of "Le Seu d'Mrgell" because it is the most emblematic building in the area.*

Inspired by the work of Luong et al. (2015b), we propose a similar but much simpler two-step approach to address the rare word problem: rather than annotating the training data with alignment information, we use unsupervised alignment models to find the sources of the unknown words in the target sentence. Once we know the source words that are responsible for the unknown target words, a word-level translation model learnt from the parallel sentences is used to translate these source words.

Our work makes the following contributions. First, we present the first study using NMT for GEC and develop a competitive NMT-based GEC system. Second, we propose a two-step approach to address the rare word problem in NMT for GEC, which we show yields a substantial improvement. Finally, we report results on several well-known publicly available test sets that can be used for cross-system comparisons.

5.2 Neural machine translation

Given a source sentence X, $X = x_1\ x_2\ ...\ x_T$, and a target sentence Y, $Y = y_1\ y_2\ ...\ y_{T'}$, where T and T' are not fixed and may be different, NMT models the conditional probability of translating the source sentence X to the target sentence Y as:

$$P(y_1, y_2, ..., y_{T'}|x_1, x_2, ..., x_T) \tag{5.1}$$

5.2.1 Recurrent neural networks

To map a variable-length input sentence to another variable-length output sentence, we use an RNN. An RNN uses a recurrent hidden state to learn sequential information and has an optional output - see Figure 5.1.

At each time step t, the recurrent hidden state h_t is calculated based on the current input x_t and the hidden state at the previous time step h_{t-1}:

$$h_t = f(x_t, h_{t-1}) \tag{5.2}$$

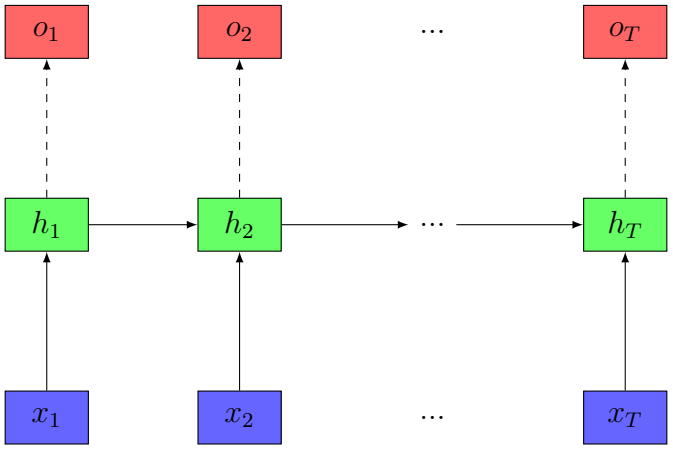


Figure 5.1: An RNN unit. The blue rectangles are the input vectors x, the red rectangles are the optional output vectors o, and the green rectangles are the hidden states h.

where f is a non-linear activation function. Traditionally, a simple sigmoid activation function is used to update the recurrent hidden state h_t:

$$h_t = \sigma(Wx_t + Uh_{t-1}) \tag{5.3}$$

where W and U are weight matrices.

Given the current state h_t, an RNN can be trained to predict a probability distribution over the next word of the sentence:

$$P(x_t|x_1, x_2, ..., x_{t-1}) = g(h_t) \tag{5.4}$$

where g is a non-linear function which outputs the probability of x_t (e.g. a softmax function - see Section 5.3).

Previous work has shown that it is difficult to train an effective RNN to capture long-distance dependencies due to the vanishing or exploding gradient problem (Bengio et al., 1994). More advanced activation functions, or recurrent units, have been proposed to better capture long-term dependencies. Two widely used ones are LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014).

LSTM

The LSTM was first proposed by Hochreiter and Schmidhuber (1997), where a memory cell and three sets of gating units were introduced - see Figure 5.2.

The activation function h_t at the time step t is defined as:

$$h_t = o_t \tanh(c_t) \tag{5.5}$$

where c_t is an *internal memory state* and o_t is an *output gate* that determines the degree to which the internal memory state will output.

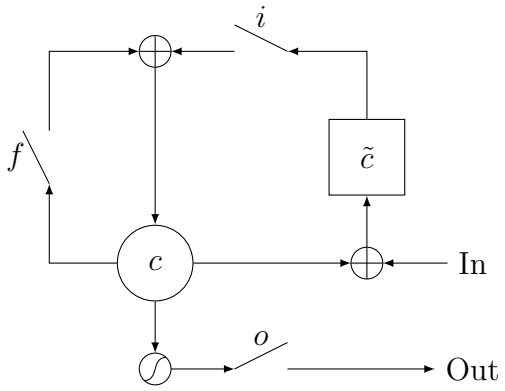


Figure 5.2: An illustration of an LSTM unit.

The *output gate* o_t is computed as:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o) \tag{5.6}$$

where σ is a logistic sigmoid function, V_o is a diagonal matrix and b_o is a bias.

The *internal memory state* c_t is updated by partially keeping the previous memory state c_{t-1} and adding a new temporary memory state $\tilde{c}_t$:

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t \tag{5.7}$$

where f_t is a *forget gate* that determines the extent to which the previous memory state c_{t-1} should be preserved and i_t is an *input gate* that decides the degree to which the new temporary memory state $\tilde{c}_t$ should be added. These two gates are computed as:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f) \tag{5.8}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i) \tag{5.9}$$

The new temporary memory state $\tilde{c}_t$ is defined as:

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1}) \tag{5.10}$$

Unlike the traditional recurrent unit whose content is overwritten at every time step (e.g. a sigmoid activation function - see Equation 5.3), the newly introduced gates enable the LSTM unit to decide whether to keep the previous memory state. Therefore, useful information collected at an early stage is more likely to be carried over a long distance, so as to capture potential long-distance dependencies.

GRU

GRU, a simplified version of LSTM, has been successfully applied to NMT (Cho et al., 2014; Bahdanau et al., 2015). Comparable results of using both GRU and

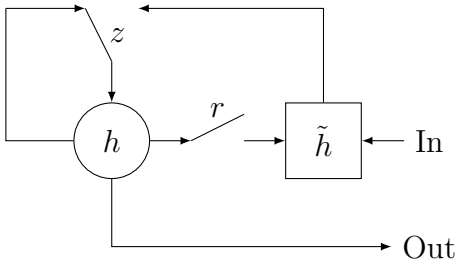


Figure 5.3: An illustration of a GRU.

LSTM on sequence modelling have been reported by Chung et al. (2014). Similar to LSTM, two new sets of gating units are introduced in GRU to modulate the flow of internal information - see Figure 5.3.

The hidden state h_t at the time step t is defined as a linear *interpolation* between the previous hidden state h_{t-1} and the new temporary hidden state $\tilde{h}_t$:

$$h_t = z_t h_{t-1} + (1 - z_t)\tilde{h}_t \tag{5.11}$$

where z_t is an *update gate* that decides the extent to which the recurrent hidden state updates itself.

The *update gate* z_t is defined as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{5.12}$$

where σ is a logistic sigmoid function.

The new temporary hidden state $\tilde{h}_t$ is defined as:

$$\tilde{h}_t = \tanh(W_h x_t + U_h r_t h_{t-1}) \tag{5.13}$$

where r_t is a *reset gate* that decides whether to forget the previous state and reset the new temporary state with the current input.

Similar to the *update gate* z_t defined is Equation 5.12, the *reset gate* r_t can be computed as:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{5.14}$$

5.2.2 Encoder-decoder

NMT applies an *encoder-decoder* framework. An encoder first reads a variable-length input sentence and encodes all (or parts) of the input sentence into a vector representation. A decoder then outputs a translation for the input sentence from the vector representation. We experiment with three different NMT models for GEC: RNN-seq2seq, BiRNN-seq2seq and Attention-seq2seq.

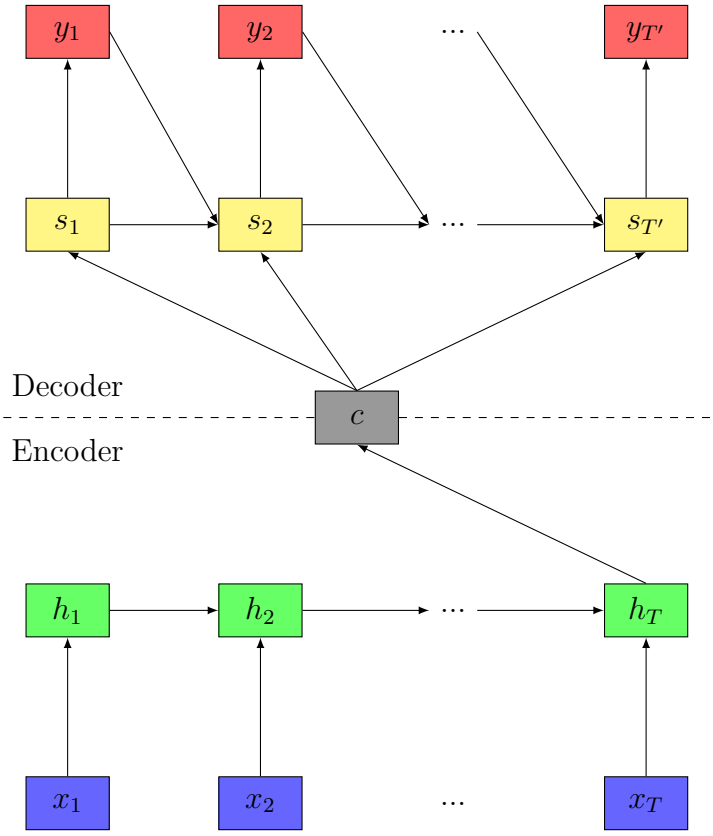


Figure 5.4: RNN-seq2seq. The blue rectangles are the input vectors x, the red rectangles are the output vectors y, the green rectangles are the encoder hidden states h, the yellow rectangles are the decoder hidden states s, and the grey rectangle is the intermediate vector c.

RNN-seq2seq

The RNN-seq2seq model was first proposed by Cho et al. (2014) and Sutskever et al. (2014) and uses two different RNNs: one as the encoder and another as the decoder - see Figure 5.4.

The RNN encoder reads an entire source sentence X into a single intermediate vector c:

$$c = q(h_T) \tag{5.15}$$

where q is a non-linear function. h_T is the final hidden state of the encoder that can be computed using Equation 5.2.

The RNN decoder then outputs a translation Y by predicting the next word y_t based on the intermediate vector c and all the previously predicted words $\{y_1, y_2, ..., y_{t-1}\}$:

$$P(Y) = \prod_{t=1}^{T'} P(y_t | \{y_1, y_2, ..., y_{t-1}\}, c) \tag{5.16}$$

and the conditional probability $P(y_t|\{y_1, y_2, ..., y_{t-1}\}, c)$ is defined as:

$$P(y_t|\{y_1, y_2, ..., y_{t-1}\}, c) = g(s_t) \tag{5.17}$$

where g is a non-linear function that outputs the probability of y_t as in Equation 5.4.

The decoder hidden state s_t at the time step t is defined as:

$$s_t = f(s_{t-1}, y_{t-1}, c) \tag{5.18}$$

As we can see, unlike the encoder hidden state h_t, the decoder hidden state s_t depends on the previous hidden state s_{t-1}, the previous output word y_{t-1} and the intermediate vector c.

BiRNN-seq2seq

In theory, an RNN is able to deal with information from any arbitrarily long sentence. However, it is more likely to capture the most recent input in practice. Sutskever et al. (2014) reported that better performance was achieved by reversing the order of the words in the input sentence. Instead of mapping (x_1, x_2, ..., x_{T-1}, x_T) to (y_1, y_2, ..., $y_{T'-1}$, $y_{T'}$), a reversed input sentence (x_T, x_{T-1}, ..., x_2, x_1) is mapped to (y_1, y_2, ..., $y_{T'-1}$, $y_{T'}$). As a result, the first few words of the output are in the immediate proximity of their original input words, for example, y_1 is in close proximity to x_1 and y_2 is close to x_2. This is equivalent to using a backward RNN encoder that reads the input sequence from the last word x_T to the first word x_1.

We propose a BiRNN-seq2seq model which consists of a new Bidirectional Recurrent Neural Network (BiRNN) (Schuster and Paliwal, 1997) encoder and the same RNN decoder used in the RNN-seq2seq model. Figure 5.5 presents the new model architecture.

The BiRNN encoder uses a forward RNN and a backward RNN. The forward RNN reads the input sentence from the first word to the last word (from x_1 to x_T), and the backward RNN reads the input sentence in reverse order (from x_T to x_1). The forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$ are calculated as:

$$\overrightarrow{h_t} = f(x_t, \overrightarrow{h_{t-1}}) \tag{5.19}$$

$$\overleftarrow{h_t} = f(x_t, \overleftarrow{h_{t+1}}) \tag{5.20}$$

A new intermediate vector c encodes both forward and backward information:

$$c = q(\overrightarrow{h_T}, \overleftarrow{h_1}) \tag{5.21}$$

where q is a non-linear function as in Equation 5.15.

Attention-seq2seq

In sequence-to-sequence problems like MT and GEC, there are some corresponding relations between the source words and the target words. Some words in the

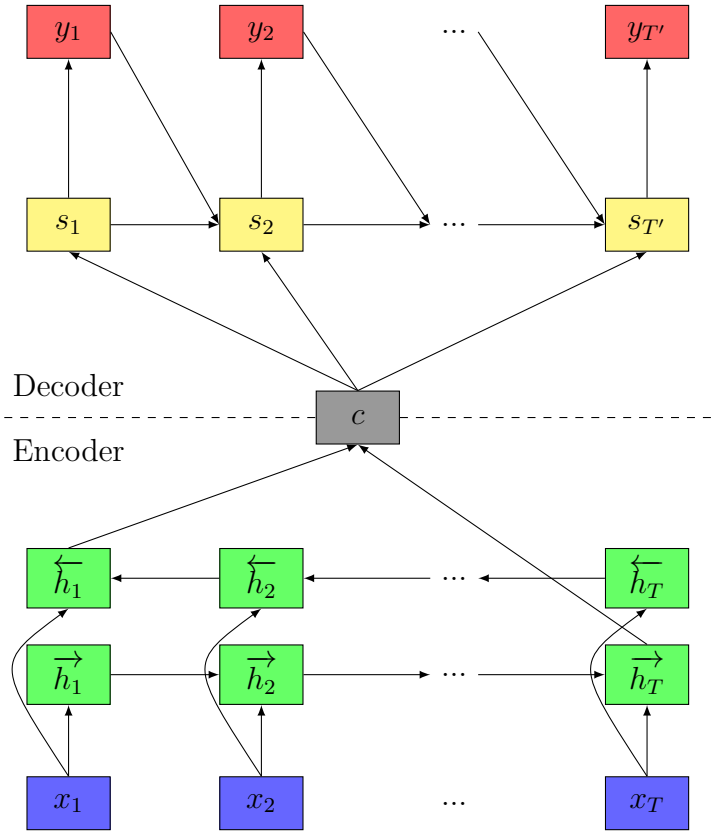


Figure 5.5: BiRNN-seq2seq. The blue rectangles are the input vectors x, the red rectangles are the output vectors y, the green rectangles are the encoder hidden states h (forward hidden states $\overrightarrow{h}$ and backward hidden states $\overleftarrow{h}$), the yellow rectangles are the decoder hidden states s, and the grey rectangle is the intermediate vector c.

input sentence might be more useful than others when predicting an output word. However, this kind of information is not used by the RNN-seq2seq model or the BiRNN-seq2seq model described above. The hidden vectors of the input words $\{h_1, h_2, ..., h_T\}$ are not directly used by the decoder; instead, the same intermediate vector c is used every time, no matter which output word the decoder attempts to predict. An attention mechanism is therefore introduced to help the decoder focus on the most relevant information in the input sentence, instead of remembering the entire input sentence. Various techniques have been proposed (Bahdanau et al., 2015; Xu et al., 2015; Luong et al., 2015a; Hermann et al., 2015). We use the soft attention mechanism described in Bahdanau et al. (2015).

Instead of using the same intermediate vector c for predicting every output word, a new vector c_t for the output word y_t at the decoding time step t is defined as:

$$c_t = \sum_{j=1}^{T} a_{tj} h_j \tag{5.22}$$

where h_j is the hidden state of word x_j in the input sentence, and a_{tj} is the

weight of h_j for predicting y_t.

By using a BiRNN encoder, h_j of word x_j can be defined by concatenating the forward hidden state $\overrightarrow{h_j}$ and the backward hidden state $\overleftarrow{h_j}$:

$$h_j = [\overrightarrow{h_j}^T; \overleftarrow{h_j}^T]^T \tag{5.23}$$

Therefore, both historical and future information is captured.

The weight a_{tj} is calculated with a softmax function:

$$a_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})} \tag{5.24}$$

and

$$e_{tj} = f(s_{t-1}, h_j) \tag{5.25}$$

where f is a feedforward neural network that calculates the score e_{tj}. The normalised weight a_{tj} can then be interpreted as the probability of the jth input word x_j being relevant to the output word y_t.

The decoder hidden state s_t is then defined using the new intermediate vector c_t:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \tag{5.26}$$

The RNN decoder outputs a translation Y:

$$P(Y) = \prod_{t=1}^{T'} P(y_t|\{y_1, y_2, ..., y_{t-1}\}, c_t) = \prod_{t=1}^{T'} g(s_t) \tag{5.27}$$

An illustration of the Attention-seq2seq model at decoding time step t is presented in Figure 5.6.

The Attention-seq2seq model can then assign different weights to different words in the input sentence, thus capturing the inherent corresponding relations between words in the input and output sentences.

5.2.3 Training an NMT system

Given a corpus of parallel sentences, an NMT system is trained to maximise log-likelihood:

$$\max_{\theta} \sum_{n=1}^{N} \log P(Y^n|X^n, \theta) = \max_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T'} \log P(y_t^n|\{y_1^n, y_2^n, ..., y_{t-1}^n\}, X^n, \theta) \tag{5.28}$$

where $\theta = [\theta_{enc}, \theta_{dec}]$ represents all the parameters, N is the total number of training examples in the corpus and (X^n, Y^n) is the nth pair.

Since both the encoder and decoder networks are differentiable with respect to their parameters θ_{enc} and θ_{dec} respectively, we maximise the log-likelihood using Stochastic Gradient Descent (SGD).

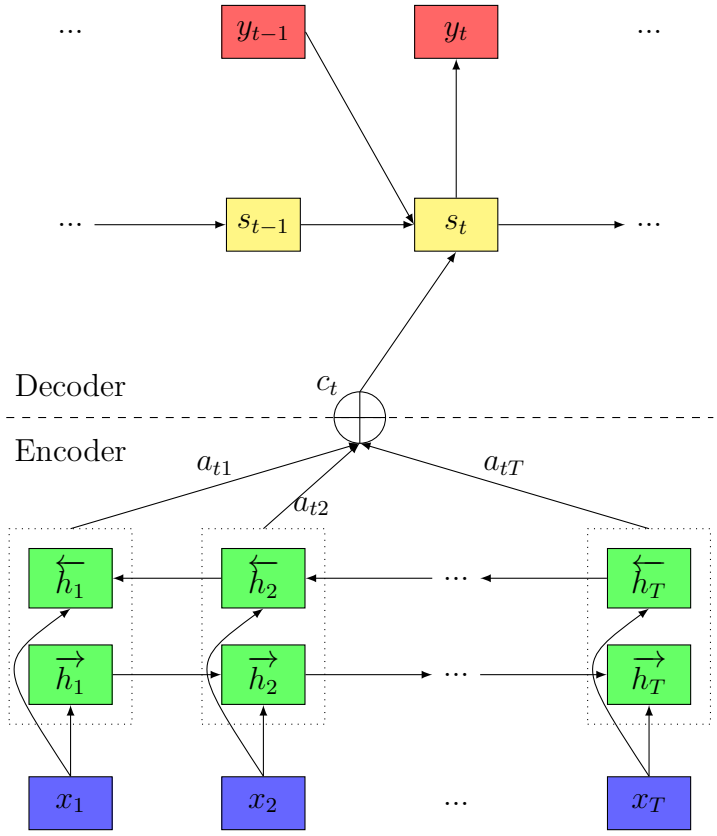


Figure 5.6: The Attention-seq2seq model at decoding time step t. The blue rectangles are the input vectors x, the red rectangles are the output vectors y, the green rectangles are the encoder hidden states h, and the yellow rectangles are the decoder hidden states s.

5.3 Handling rare words

NMT suffers from the rare word problem. When predicting a target word, we use a multilayer network (Pascanu et al., 2014) with a single maxout hidden layer (Goodfellow et al., 2013) and normalise the output probabilities of every target word with a softmax function:

$$P(y_t|\{y_1, y_2, ..., y_{t-1}\}, X) = \frac{\exp(w_t^T s_t + b_t)}{\sum_{k:y_k \in V} \exp(w_k^T s_t + b_k)} \tag{5.29}$$

where w and b are the target word vector and bias respectively, and V is the set of all the target words.

Due to the computational complexity of the softmax function in Equation 5.29, NMT systems often use a shortlist of 30,000 to 80,000 most frequent words. Any word not included in the shortlist is replaced by the *UNK* symbol. Therefore, NMT systems are not capable of translating rare words that are not included in the shortlist. This harms translation quality, and Sutskever et al. (2014) and Bahdanau et al. (2015) have shown that their NMT systems produced much worse BLEU scores on sentences with rare words.

Phrase-based SMT systems, like the one we developed in Chapter 3, do not have the same rare word problem. Our phrase-based SMT system uses a phrase translation table, which contains all the phrase alignments learnt from the training data. Phrase alignments in the table are then used directly by the SMT decoder during translation. Unlike NMT, SMT systems do not need to limit their vocabulary sizes during training, and all the words present in the training data are used to learn phrase alignments. During testing, new words that have not been seen in the training data remain unchanged.

Two different approaches have been proposed to address the rare word problem in NMT. Luong et al. (2015b) introduced three new annotation strategies to annotate the training data, so that unknown words in the output can be traced back to their origins. They used word alignment algorithms to re-annotate the training data and built NMT systems based on the re-annotated data. Information about the unknown words in the target sentence and their corresponding words in the source sentence were extracted. In a post-processing step, the unknown words were translated using a dictionary. Jean et al. (2015a) proposed an approximate training algorithm based on importance sampling, which approximates softmax functions by selecting only a small subset of the target vocabulary. They have shown that an NMT system can be trained with a very large target vocabulary without increasing the training complexity. These two approaches are complementary and can be combined together to yield further improvements.

We propose a similar but much simpler two-step approach to perform *UNK* replacement: 1) align the unknown words (i.e. *UNK* tokens) in the target sentence to their origins in the source sentence using unsupervised aligners directly; 2) build a word-level translation model to translate those words in a post-processing step. Due to the nature of error correction (i.e. both source and target sentences are in the same language), most words translate as themselves, and errors are often similar to their correct forms. Therefore, we hypothesise that unsupervised aligners can be used effectively to align the unknown target words. Our *UNK* replacement approach is different from the one proposed in Luong et al. (2015b) in that: 1) we avoid re-annotating any training data; 2) we use only the NMT system output; and 3) we apply unsupervised aligners directly to locate the source words that are responsible for the unknown target words. Our approach is also different from the one proposed in Jean et al. (2015a) as we treat the NMT system as a black box, therefore our approach can be used with any NMT system.

We use two automatic alignment tools: GIZA++ and METEOR, which have already been used in Chapter 3 (Section 3.3.1) and Chapter 4 (Section 4.4.5.2). GIZA++ is an implementation of IBM Models 1-5 and the HMM, which aligns two sentences from any pair of languages. Unlike GIZA++, METEOR aligns two sentences from the same language by identifying not only words with exact matches, but also words with identical stems, synonyms, and unigram paraphrases. To build a word-level translation model for translating the source words that are responsible for the target unknown words, we need word-aligned data. IBM Models are used to learn word alignments from the parallel training data. For words that have not been seen in the training data, we keep the source words unchanged.

5.4 Experiments

5.4.1 Experimental set-up

We follow the experimental set-up described in Section 4.4.1. We use the publicly available FCE dataset, and extract additional training examples from the CLC. Training and test data is pre-processed using RASP. System performance is evaluated using the I-measure.

5.4.2 Training details

We use GroundHog,[4] a python framework on top of Theano[5] that provides a flexible and efficient way of implementing complex RNN models. All our models are trained using graphics processing units (GPUs).[6]

The GRU is used as the activation function in RNNs. The initial parameter settings follow previous work (see Cho et al., 2014; Bahdanau et al., 2015; Jean et al., 2015a). We use a hidden size of 1,000 for the RNN layer and the feedforward neural network layer. We set the size of the maxout hidden layer to 500. The dimensionality of word embeddings is 620. We limit the source and target vocabulary to the most frequent 30,000 words and replace any rare word with the *UNK* token.

Our models are trained with mini-batch SGD using the Adadelta algorithm (Zeiler, 2012) with hyper-parameters $\epsilon = 10^{-6}$ and $\rho = 0.95$. Gradients are clipped at 1 to alleviate the exploding gradient problem as suggested by Pascanu et al. (2013). We apply dropout (Srivastava et al., 2014) at a rate of 0.5 to feed-forward connections in RNNs. As the mini-batch size is subject to a memory limit, we reduce the mini-batch size when training large models (e.g. with a larger vocabulary, using longer sentences or having more hidden units). For example, when training models with the sentences of length up to 30, 50, 80 and 100 tokens, we use a mini-batch of 80, 40, 30 and 30 sentences respectively. However, having a small mini-batch size may result in a noisy update because the gradient is averaged over the mini-batch. We therefore allow the models using a small mini-batch size to train for more iterations. Weight parameters are initialised by sampling from a white Gaussian distribution ($\mu = 0$ and $\sigma = 0.01$) and biases are initialised to zero.

We use a *beam search* to find a correction that approximately maximises the conditional probability. During *beam search*, we keep a beam size of 10 and discard all other hypotheses.

For our experiments, we first compare three different NMT models; then vary the sentence length, the beam size and the vocabulary size; finally, replace *UNK* tokens using our proposed two-step approach.

[4] `https://github.com/lisa-groundhog/GroundHog`
[5] `http://deeplearning.net/software/theano`
[6] Tesla K20 and Titan X

Model	Sentence length	Training time	WAcc (%)	I (%)
Baseline	-	-	86.83	0
SMT	-	-	87.21	2.87
RNN-seq2seq	30	19 hours	72.37	-16.66
	50	28 hours	77.07	-11.24
	80	44 hours	77.46	-10.79
	100	121 hours	78.08	-10.08
BiRNN-seq2seq	30	25 hours	76.43	-11.98
	50	42 hours	78.56	-9.83
	80	51 hours	79.05	-9.05
	100	76 hours	79.46	-8.49
Attention-seq2seq	30	30 hours	85.06	-2.04
	50	52 hours	85.34	-1.72
	80	64 hours	85.49	-1.54
	100	82 hours	85.71	-1.30

Table 5.1: Performance of three NMT models with different sentence lengths on the FCE test set.

5.4.3 NMT models

We build NMT systems using the RNN-seq2seq, BiRNN-seq2seq and Attention-seq2seq models. For each model, we train four systems with sentences of length up to 30, 50, 80 and 100 tokens. System training time and results are presented in Table 5.1. *Baseline* is a baseline system which makes no corrections and *SMT* is the best SMT system from Section 4.4.2. We can see that NMT systems are not able to achieve comparable results to the SMT system. Negative I scores suggest that NMT systems seem to make the source sentences worse. Among three NMT models, systems built using the RNN-seq2seq model yield the worst I scores. A closer observation of the system output reveals a large number of unnecessary changes introduced by the systems. Adding a backward RNN layer helps, as systems using the BiRNN-seq2seq model outperform those using the RNN-seq2seq model. Systems trained with the Attention-seq2seq model achieve much better I scores. As discussed earlier in Section 5.2.2, the RNN-seq2seq and BiRNN-seq2seq models are unable to use the corresponding relations between the source words and the target words. This kind of information is used by the Attention-seq2seq model, therefore, systems using the Attention-seq2seq model are more likely to keep error-free source words untouched and only make necessary changes. As the Attention-seq2seq model produces scores that are close to the SMT system, we believe there is room for improvement and decide to use it in later experiments.

5.4.4 Sentence length

In Table 5.1, we can also see that systems trained with longer sentences outperform systems trained with shorter sentences for all three NMT models. Systems trained

Sentence length	No.	Prop.
30	1,815,051	92.33%
50	1,947,350	99.07%
80	1,963,685	99.90%
100	1.964,992	99.96%
All	1,965,727	100.00%

Table 5.2: Data coverage for different sentence lengths on the training set.

with sentences of length up to 100 tokens yield the best performance, followed by 80, 50 and 30. This is probably because increasing the sentence length limit from 30 to 100 causes more examples to be added to our training set (see Table 5.2). When we limit the sentence length to 30 tokens, 92.33% of all the training examples are used, while increasing it to 100 tokens pushes the percentage up to 99.96%. During training, we also reduce the mini-batch size accordingly as discussed in Section 5.4.2. We can see that it takes more time to train models with longer sentences and smaller mini-batch sizes.

5.4.5 Beam size

For the four NMT systems built using the Attention-seq2seq model (i.e. with sentence lengths at 30, 50, 80 and 100), we vary the beam size between 5 and 100 for decoding. Results are presented in Table 5.3. We can see that increasing the decoder's beam size does not yield a consistent improvement in system performance, but increases the decoding time. Using a beam size of 10 yields the best I scores for sentence lengths 30, 80 and 100, so we keep the beam size at 10 for all models.

5.4.6 Vocabulary size

The source side of our training data contains 28,823,615 words in total with 248,028 unique words while the target side contains a total of 29,219,128 words with 143,852 unique words. As we can see, the source side vocabulary size is much larger than that of the target side as there are many incorrect words in the source (e.g. spelling mistakes and word form errors). We thus aim to investigate the effect of vocabulary by experimenting with different source and target vocabulary sizes. The source vocabulary size is selected from $\{30k, 50k, 80k, 100k, 150k\}$ and the target vocabulary size is selected from $\{30k, 50k, 80k\}$. Table 5.4 presents data coverage for different vocabulary sizes on the source and target sides. *All* refers to all the tokens in the source side of the training data. As we can see that there are still unknown test words even if we cover all the words in the training set.

Results of using different source and target vocabulary sizes are presented in Table 5.5. Our experiments show that using a large vocabulary size helps and that increasing the source side vocabulary size is more useful than target side. In particular, increasing the source vocabulary size yields a consistent improvement in system performance. When we limit the target vocabulary size to 30k, systems trained with

Sentence length	Beam size	Decoding speed	WAcc (%)	I (%)
30	5	0.23s/sentence	85.05	-2.05
	10	0.34s/sentence	**85.06**	**-2.04**
	20	0.50s/sentence	85.05	-2.05
	30	0.66s/sentence	85.04	-2.06
	40	0.86s/sentence	85.04	-2.06
	50	1.10s/sentence	85.02	-2.08
	100	1.91s/sentence	85.01	-2.10
50	5	0.23s/sentence	85.33	-1.73
	10	0.34s/sentence	85.34	-1.72
	20	0.50s/sentence	85.36	-1.70
	30	0.66s/sentence	85.36	-1.69
	40	0.86s/sentence	**85.37**	**-1.69**
	50	1.10s/sentence	**85.37**	**-1.69**
	100	1.91s/sentence	**85.37**	**-1.69**
80	5	0.23s/sentence	85.47	-1.57
	10	0.34s/sentence	**85.49**	**-1.54**
	20	0.50s/sentence	85.49	-1.55
	30	0.66s/sentence	85.48	-1.55
	40	0.86s/sentence	85.48	-1.56
	50	1.10s/sentence	85.47	-1.56
	100	1.91s/sentence	85.47	-1.56
100	5	0.23s/sentence	**85.71**	**-1.30**
	10	0.34s/sentence	**85.71**	**-1.30**
	20	0.50s/sentence	85.67	-1.34
	30	0.66s/sentence	85.60	-1.42
	40	0.86s/sentence	85.56	-1.47
	50	1.10s/sentence	85.56	-1.47
	100	1.91s/sentence	85.56	-1.47

Table 5.3: Results of the Attention-seq2seq model with different decoder beam sizes and sentence lengths on the FCE test set. The best results for each sentence length are marked in **bold**.

a source vocabulary size of 150k perform the best (*150k-30k*[7] in Table 5.5), followed by 100k (*100k-30k*), 80k (*80k-30k*), 50k (*50k-30k*) and 30k (*30k-30k*). For systems with a target vocabulary size of 50k, the *80k-50k* group outperforms the *50k-50k* group (except for systems with sentence length at 100). However, we do not observe a similar consistent improvement when increasing the target vocabulary size since, for example, systems trained on *80k-80k* produce worse I scores than those trained on *80k-50k*. As we can see, the performance of the current best NMT system (i.e. *NMT 150k-30k* with sentence length at 100) is still worse than a 'do-nothing' baseline and our best SMT system.

[7][source vocabulary size] - [target vocabulary size]

Vocabulary size	Source side		Target side	
	Train	Test	Train	Test
30k	98.6	98.4	99.4	99.3
50k	99.1	98.7	99.6	99.5
80k	99.3	99.0	99.8	99.5
100k	99.5	99.1	99.8	99.6
150k	99.7	99.1	100.0	99.6
All	100.0	99.3	-	-

Table 5.4: Data coverage for different vocabulary sizes on the source and target side (in percentages).

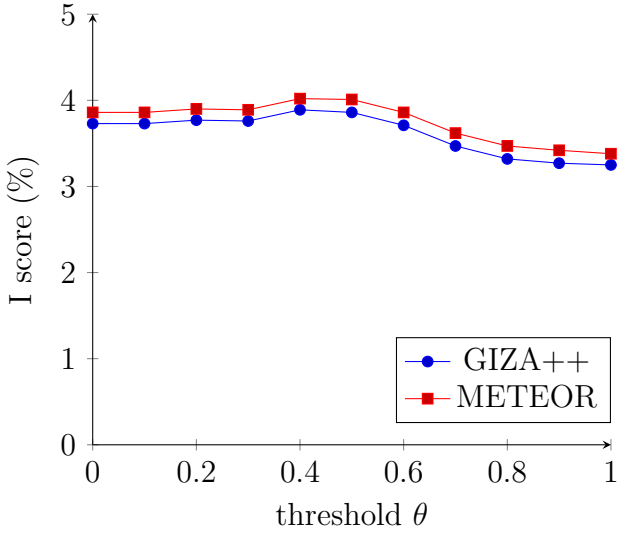


Figure 5.7: Results of using different thresholds on the FCE test set for *NMT 30k-30k*.

5.4.7 *UNK* replacement

We have observed that systems trained with longer sentences outperform those trained with shorter sentences. Therefore, we select systems trained with sentences of length up to 100 tokens and replace all the *UNK* tokens in their output. GIZA++ and METEOR are used to align the *UNK* tokens to their source words. We build an additional word-level TM from all the parallel examples in our training set.[8] *UNK* tokens in the NMT output are replaced by the translations of their source words from the new TM.

During translation, we set a threshold θ and only apply translation mappings with probability scores above the threshold. For words with translation probability scores lower than θ or with no translations in the TM, we keep the source words unchanged. We first experiment with the *NMT 30k-30k* model and find out that setting the threshold θ to 0.4 yields the best performance, as shown in Figure 5.7.

[8]There is no need to limit the source and target vocabulary when training the new word-level TM (as in SMT).

Vocabulary size		Sentence length	Training time	WAcc (%)	I (%)
Source	Target				
30k	30k	30	30 hours	85.06	-2.04
		50	52 hours	85.34	-1.72
		80	64 hours	85.49	-1.54
		100	82 hours	85.71	-1.30
50k	30k	30	45 hours	85.43	-1.62
		50	79 hours	85.66	-1.35
		80	52 hours	85.83	-1.15
		100	69 hours	85.84	-1.14
80k	30k	30	38 hours	85.13	-1.96
		50	65 hours	85.70	-1.30
		80	89 hours	85.87	-1.11
		100	105 hours	85.93	-1.04
100k	30k	30	29 hours	85.15	-1.94
		50	53 hours	85.82	-1.16
		80	91 hours	85.90	-1.07
		100	118 hours	85.96	-1.01
150k	30k	30	46 hours	85.21	-1.87
		50	58 hours	85.97	-1.00
		80	89 hours	85.98	-0.98
		100	95 hours	**86.15**	**-0.79**
50k	50k	30	53 hours	84.88	-2.24
		50	44 hours	85.44	-1.60
		80	73 hours	85.73	-1.27
		100	57 hours	85.80	-1.18
80k	50k	30	53 hours	85.42	-1.63
		50	60 hours	85.66	-1.34
		80	119 hours	85.76	-1.23
		100	120 hours	85.72	-1.28
80k	80k	30	54 hours	85.05	-2.05
		50	105 hours	85.10	-1.99
		80	107 hours	85.15	-1.94
		100	121 hours	85.62	-1.40

Table 5.5: Results of the Attention-seq2seq model using different source and target vocabulary sizes and sentence lengths on the FCE test set. The best results are marked in **bold**.

Results of NMT systems with *UNK* replacement are presented in Table 5.6. When we replace the *UNK* tokens in the NMT output, using GIZA++ for unknown word alignment improves the system performance for all NMT systems. The introduction of the METEOR alignment information to GIZA++ yields further improvements. We can see that our *UNK* replacement approach is effective and provides a

Model	Vocabulary size		*UNK* replacement	WAcc (%)	I (%)
	Source	Target			
Baseline	-	-	-	86.83	0
SMT	-	-	-	87.21	2.87
NMT	30k	30k	None	85.71	-1.30
			GIZA++	**87.34**	**3.89**
			GIZA++ & METEOR	**87.36**	**4.02**
	50k	30k	None	85.84	-1.14
			GIZA++	87.21	2.87
			GIZA++ & METEOR	**87.22**	**2.89**
	80k	30k	None	85.93	-1.04
			GIZA++	87.14	2.37
			GIZA++ & METEOR	87.15	2.40
	100k	30k	None	85.96	-1.01
			GIZA++	87.18	2.63
			GIZA++ & METEOR	87.19	2.70
	150k	30k	None	86.15	-0.79
			GIZA++	**87.49**	**5.00**
			GIZA++ & METEOR	**87.50**	**5.06**
	50k	50k	None	85.80	-1.18
			GIZA++	87.02	0.96
			GIZA++ & METEOR	87.03	0.99
	80k	50k	None	85.72	-1.28
			GIZA++	86.77	-0.07
			GIZA++ & METEOR	86.78	-0.03
	80k	80k	None	85.62	-1.40
			GIZA++	86.66	-0.19
			GIZA++ & METEOR	86.67	-0.15

Table 5.6: Results of *UNK* replacement on the FCE test set. Improvements over the SMT system are marked in **bold**.

substantial improvement. All NMT systems produce positive I scores - improving the original sentence quality - after *UNK* replacement except *NMT 80k-50k* and *NMT 80k-80k*. Three NMT systems even outperform our best SMT system after *UNK* replacement: *NMT 150k-30k* (achieves an I score of 5.06% using GIZA++ & METEOR and an I score of 5.00% using GIZA++), *NMT 30k-30k* (achieves an I score of 4.02% using GIZA++ & METEOR and an I score of 3.89% using GIZA++) and *NMT 50k-30k* (achieves an I score of 2.89% using GIZA++ & METEOR).

5.5 Analysis and discussion

Comparing the output of our best SMT system with that of our best NMT system reveals that some errors that are missed by the SMT system are captured by the

System	GLEU	$F_{0.5}$	I
Baseline	59.14	0	0
SMT	60.90	28.30	-2.60
SMT + SVM	61.12	27.86	-1.65
Our NMT systems			
NMT 150k-30k	60.49	26.94	-3.94
NMT 30k-30k	60.65	29.13	-4.11
NMT 50k-30k	60.53	29.15	-4.33

Table 5.7: Results of the baseline, the SMT system, the best SVM re-ranker and our NMT systems on the CoNLL-2014 development set (in percentages).

NMT system. As discussed before, our phrase-based SMT system is trained on surface forms and it has to have seen the exact correction mapping in the training data in order to make a correction. Since the NMT system does not rely on any correction mappings, in theory, it should be able to make any changes as long as it has seen the words in the training data. For example:

Example 5.5. Missed *RN* (Replace Noun):

Original sentence	*You can find a lot of **documentary** about it and you have several competitors personal objects.*
Smt output	*You can find a lot of **documentary** about it and you have several competitors personal objects.*
Nmt output	*You can find a lot of **documents** about it and you have several competitors personal objects.*
Gold standard	*You can find a lot of **documents** about it and you have several competitors personal objects.*

The SMT system fails to correct the noun error as the correction mapping *documentary* → *documents* is not in the SMT phrase table learnt from the training data. However, as these two words 'documentary' and 'documents' have been seen in the training data, the NMT system is able to successfully detect and correct the error.

5.5.1 Results on the CoNLL-2014 shared task development set

As in Section 4.5.1, we apply our NMT-based GEC systems trained on the CLC to the CoNLL-2014 shared task development set. We select three NMT systems that outperform the SMT system in Section 5.4.7: *NMT 150k-30k*, *NMT 30k-30k* and *NMT 50k-30k*. System performance is evaluated using GLEU, $F_{0.5}$ and I-measure (see Table 5.7). *Baseline* is a baseline system which makes no corrections, *SMT* is the final SMT system from Chapter 3 (i.e. the one used in our winning system submitted to the CoNLL-2014 shared task) and *SVM* is the best SVM re-ranker from Chapter 4. Results show that our NMT systems produce worse GLEU and I scores, but better $F_{0.5}$ scores. The *NMT 50k-30k* system yields the best $F_{0.5}$ score.

Type performance for the *NMT 50k-30k* system and the SMT system is given in Table 5.8. We can see that the NMT system is better at correcting *ArtOrDet*, *Nn*, *Pform*, *Pref*, *Prep*, *SVA*, *Spar*, *Srun*, *Um*, *Vform* and *Wtone* errors than the SMT system. While the NMT system is able to correct some types of errors that are completely missed by the SMT system (e.g. *Srun*, *Um* and *Wtone*), it is incapable of correcting certain types of errors that can be corrected by the SMT system (such as *Others* and *Woinc*). Both our SMT and NMT systems fail to correct *Cit*, *Sfrag*, *Smod*, *Woadv* and *Wa* errors. Our analysis reveals that corrections from the SMT and NMT systems are complementary.

In the following examples, we show some cases where the NMT system corrects errors that are missed by the SMT system:

Example 5.6. *Wtone*:

Original sentence	*Thus, let **'s** us discuss the pros and cons ...*
Smt output	*Thus, let **'s** us discuss the pros and cons ...*
Nmt output	*Thus, let us discuss the pros and cons ...*
Gold standard	*Thus, let us discuss the pros and cons ...*

Example 5.7. *Wform*:

Original sentence	*There are **kidnaps** everywhere and not all of the **family** can afford the ransom ...*
Smt output	*There are **kidnaps** everywhere and not all of the **families** can afford the ransom ...*
Nmt output	*There are **kidnappings** everywhere and not all of the **families** can afford the ransom ...*
Gold standard	*There are **kidnappings** everywhere and not all of the **families** can afford the ransom ...*

5.5.2 Results on the CoNLL-2014 shared task test set

Similar to Section 4.5.2 and in order to test how well our system generalises, we apply our NMT systems trained on the CLC to the CoNLL-2014 shared task test data directly without adding the NUCLE data or tuning for $F_{0.5}$.

We compare our NMT systems with the top three systems in the shared task. Evaluation is performed using GLEU, $F_{0.5}$ and I-measure on the original test set and presented in Table 5.9. As we can see, our *NMT 150k-30k* and *NMT 30k-30k* systems outperform the top three teams on all evaluation metrics even though our systems are not trained on the NUCLE data. The *NMT 150k-30k* system achieves the best I score (-2.88%), while the *NMT 30k-30k* system achieves the best GLEU (65.59%) and the best $F_{0.5}$ (39.90%). These results show that our NMT-based GEC systems generalise well.

Error type	SMT			NMT		
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
ArtOrDet	49.49	21.08	38.98	**53.69**	**26.49**	**44.54**
Cit	-	0.00	0.00	-	0.00	0.00
Mec	62.26	18.13	41.88	**71.05**	14.84	40.42
Nn	50.52	24.87	41.88	**67.29**	**36.55**	**57.60**
Npos	8.70	6.90	8.26	7.14	6.90	7.09
Others	25.00	5.71	14.93	0.00	0.00	0.00
Pform	9.09	3.85	7.14	**22.22**	**7.69**	**16.13**
Pref	33.33	2.56	9.80	**100.00**	2.56	**11.63**
Prep	38.16	9.35	23.62	31.82	**13.55**	**25.06**
Reordering	0.00	-	0.00	0.00	-	0.00
Rloc-	41.67	7.81	22.32	33.33	1.56	6.58
SVA	33.33	10.17	22.90	**59.26**	**27.12**	**47.90**
Sfrag	-	0.00	0.00	-	0.00	0.00
Smod	-	0.00	0.00	-	0.00	0.00
Spar	100.00	5.88	23.81	100.00	**11.76**	**40.00**
Srun	0.00	0.00	0.00	**44.44**	**7.27**	**21.98**
Ssub	33.33	4.23	14.02	22.22	2.82	9.35
Trans	21.74	3.55	10.73	20.83	3.55	10.55
Um	-	0.00	0.00	**50.00**	**2.94**	**11.90**
V0	25.00	5.56	14.71	20.00	5.56	13.16
Vform	32.76	16.10	27.14	**42.31**	**18.64**	**33.74**
Vm	50.00	8.42	25.16	11.76	4.21	8.66
Vt	37.14	8.39	22.03	15.09	5.16	10.90
Woadv	-	0.00	0.00	-	0.00	0.00
Woinc	3.85	5.41	4.08	0.00	0.00	0.00
Wa	0.00	0.00	0.00	0.00	0.00	0.00
Wci	13.27	3.05	8.52	4.03	2.80	3.70
Wform	58.62	22.57	44.43	43.31	**24.34**	37.47
Wtone	-	0.00	0.00	**100.00**	**7.69**	**29.41**
TOTAL	39.58	13.23	28.30	36.45	**16.18**	**29.15**

Table 5.8: Type-specific M^2 performance of the SMT system and the NMT system on the CoNLL-2014 development set (in percentages). NMT improvements over the SMT system are marked in **bold**.

5.6 Recent work

To address the rare word problem in NMT, apart from the two approaches based on word-based NMT models (Luong et al., 2015b; Jean et al., 2015a), it is possible to use models that work with smaller units. Based on the intuition that various word classes are translatable via units that are smaller than words, Sennrich et al. (2016) introduced subword models that encode rare and unknown words as sequences of

System	GLEU	$F_{0.5}$	I
Baseline	64.19	0	0
Our NMT systems			
NMT 150k-30k	65.47	38.25	-2.88
NMT 30k-30k	65.59	39.90	-3.11
NMT 50k-30k	63.92	34.53	-4.11
Top 3 systems in CoNLL-2014			
CAMB (Felice et al., 2014)	64.32	37.33	-5.58
CUUI (Rozovskaya et al., 2014a)	64.64	36.79	-3.91
AMU (Junczys-Dowmunt and Grundkiewicz, 2014)	64.56	35.01	-3.31

Table 5.9: System performance on the CoNLL-2014 test set without alternative answers (in percentages).

subword units. Ling et al. (2015) and Costa-Jussà and Fonollosa (2016) proposed the use of character-based NMT models where the source and target sentences are seen as sequences of characters rather than words. Instead of using word embeddings in word-based NMT models, Ling et al. (2015) introduced a *character-to-word* compositional model while Costa-Jussà and Fonollosa (2016) used character-based embeddings in combination with convolutional and highway layers.

We notice that similar work on using NMT for error detection and correction has recently been done by Xie et al. (2016) and Schmaltz et al. (2016). Xie et al. (2016) employed a similar RNN *encoder-decoder* framework to build a GEC system. Unlike in our *UNK* replacement approach, they used a character-based model to handle rare words and spelling mistakes. An additional LM was used during decoding and a multilayer perceptron binary classifier was built to filter out unnecessary changes made by the NMT model (similar to the one developed by Hoang et al. (2016)). Their final system achieved an $F_{0.5}$ score of 40.56% on the CoNLL-2014 test set.

Schmaltz et al. (2016) used NMT models for sentence-level grammatical error identification. By combining three character-based *encoder-decoder* models, one word-based model and a sentence-level CNN, they produced the best performing system on the 2016 Automated Evaluation of Scientific Writing binary prediction shared task (Daudaravicius et al., 2016). Instead of mapping a source sentence to its corrected version, the authors paired it with its literal annotation, e.g.

Example 5.8. In the training data:

INPUT *The models works .*
OUTPUT *The models <del> works </del> <ins> work </ins> .*

Chollampatt et al. (2016) made use of continuous vector representations in a different way. They investigated the effectiveness of two neural network TMs: a neural network global lexicon model (Ha et al., 2014) and a neural network joint model (Devlin et al., 2014), showing that they can improve the performance of an SMT-based GEC system. Their system achieved an $F_{0.5}$ score of 41.75% on the CoNLL-2014 test set.

5.7 Summary

In this chapter, we have investigated NMT for GEC. We have proved that NMT can be successfully applied to GEC once we address the rare word problem. We have compared three different NMT models (RNN-seq2seq, BiRNN-seq2seq and Attention-seq2seq) and shown that a BiRNN is effective while an attention mechanism is crucial to help the system keep error-free source words unchanged and only make necessary changes. We have also shown that systems trained on longer sentences (and/or probably a larger number of sentences) perform better. Using a large vocabulary size is also helpful, particularly on the source side. Our proposed two-step approach for *UNK* replacement has been proved to be effective and provide a substantial improvement. We have developed an NMT-based GEC system that generalises well to other datasets. Our NMT system achieves an I score of 5.06% on the publicly available FCE test set, outperforming our best SMT system with an I score of 2.87%. When testing on the official CoNLL-2014 test set without alternative answers, our system outperforms the top three teams in the shared task.

chapter 6

Conclusion

This thesis has focussed on GEC for non-native English text. We have treated it as a translation task from incorrect into correct English, developed three main variants of end-to-end all-errors GEC systems and explored many contrasting parameterisations of these models.

In Chapter 3, we investigated SMT for building an all-errors GEC system. We first identified issues that arise from applying existing SMT to GEC, and then proposed solutions to address some of the issues. We presented the development of an SMT-based GEC system, which forms one half of our winning system submitted to the CoNLL-2014 shared task. The winning system, according to Bryant and Ng (2015), was able to perform 73% as reliably as a human annotator when further alternative corrections are taken into account.

Results from our SMT-based GEC system were analysed and discussed in depth. A detailed analysis of system performance by type was also presented. This kind of analysis is valuable as it helps us better understand the strengths and weaknesses of the system, as well as diagnose problems and identify areas for future improvement. Our findings suggest that an SMT-based GEC system is particularly good at correcting errors that have more training examples, involve changes of only one or a few words and depend on local context. When looking at error types, the system achieves the best performance for *Wform*, *Mec*, *Nn* and *ArtOrDet*. Our results also confirm that the SMT-based GEC system is able to correct sequential errors and interacting errors in one go. Forced decoding experiments reveal that about 54% of all missed errors are due to SMT decoding errors, as better corrections were observed in the candidate pool produced by the SMT system but the decoder failed to select them. The remaining 46% missed errors are OOV errors, since the needed correction mappings could not be learnt from the training data. The next two chapters described attempts to solve these problems.

Chapter 4 addressed SMT decoding errors via candidate re-ranking. Since SMT was not originally designed for error correction, we argued that it is necessary to add new features that are tailored for GEC to help the SMT decoder better distinguish good from bad corrections. We proposed a supervised ranking model to re-rank candidates generated by an SMT-based GEC system. To the best of our knowledge, we are the first to use a supervised discriminative re-ranking model in SMT for

GEC, showing that n-best list re-ranking can improve sentence quality. A range of novel linguistic features were investigated and implemented in our re-ranker. We developed an SVM re-ranker which was proved to be effective in re-ranking correction candidates for GEC and generalise well to different corpora.

Future work includes the optimisation of the n-best list size, which is one of the most effective parameters in re-ranking. Additionally, we would like to explore more discriminative features. Syntactic features may provide useful information to correct potentially long-distance errors, such as those involving agreement. Mizumoto and Matsumoto (2016) showed that shallow syntactic features based on POS and parse tags are effective. Future work includes investigating other types of syntactic features and comparing them with the non-syntactic features used in our current re-ranker. We may also need features to capture the semantic similarity between the source and target sentences as retaining the meaning of the source sentence after correction is important. Neural LMs and TMs may additionally help capture syntactic and semantic information. It is also worth trying GEC re-ranking jointly for larger context, as corrections for some errors may require a signal outside the sentence boundaries, for example by adding new features computed from surrounding sentences.

Chapter 5 addressed data sparsity and SMT OOV errors using more general neural network models. This constitutes the first study on NMT for GEC. For errors whose correction mappings have not been seen in the training data, we hypothesised that NMT-based GEC systems may have a chance to correct them, given the fact that NMT does not rely on any correction mappings and the use of distributed representations for words helps alleviate the *curse of dimensionality*. We addressed problems from adapting the existing NMT framework to GEC. In particular, we proposed a two-step *UNK* replacement approach to handle the rare word problem, which has been proved to be effective and provide a substantial improvement. The results of our experiments confirm that NMT can be successfully applied to GEC and that an NMT-based GEC system is able to correct some of the errors that are missed by an SMT-based system.

Due to time limitations, we have only presented a tentative study on NMT for GEC and so have not yet been able to exploit its full potential. However, we hope the positive results we have demonstrated in this thesis will encourage further research on the adaptation of NMT to GEC. In the future, we would like to explore other ways of addressing the rare word problem in NMT-based GEC, such as incorporating the alignment information generated by the attention-based decoder or using models that work with smaller units. For example, a character-based model was used by Xie et al. (2016) and Schmaltz et al. (2016) to handle rare words and spelling mistakes. We expect further improvement by combining the character-based model with our *UNK* replacement approach. Li and Jurafsky (2016) showed that re-ranking the n-best list from an NMT system with additional information (e.g. maximum mutual information and sentence length) yielded consistent improvements for MT tasks, while Xie et al. (2016) observed an increase in system performance after adding a LM to their NMT-based GEC system. Therefore, it is worth investigating ways to help our NMT system output better correction candidates, for example by using the candidate re-ranking techniques developed in Chapter 4. In addition to building

stand-alone NMT systems, previous work on MT has shown that NMT models help SMT re-ranking. Improvements over state-of-the-art SMT systems were observed when using NMT to re-rank the output of phrase-based SMT systems (Neubig et al., 2015). Another area for further research is to use NMT features to help candidate re-ranking for SMT-based GEC. Our analysis reveals that the NMT system is capable of capturing some of the errors that are missed by the SMT system, and that corrections made by the two systems are complementary. How to effectively combine the corrections from the SMT and NMT systems remains a problem.

With reference to the aims of this thesis described in Section 1.2, we can now answer the three research questions: we have shown that SMT can form the basis of a competitive all-errors GEC system, SVM re-ranking can improve sentence quality in SMT-based GEC and NMT can be successfully applied to GEC to capture errors missed by SMT-based GEC.

Due to the success of our early work, more people have started to use SMT for GEC and claimed better results. Among them, Junczys-Dowmunt and Grundkiewicz (2016) developed an SMT-based GEC system and reported state-of-the-art M^2 performance on the CoNLL-2014 shared task test set. After introducing new features and models, they tuned the system on the NUCLE data towards the M^2 metric. Despite their promising result, we believe it largely depends on the training/tuning data, newly introduced features and tuning metrics. There is no doubt that parameter tuning can be very effective, but we have focused on building more appropriate and generalised models for GEC in this thesis. As these two research directions are complementary to each other, we could replicate their experiments and expect better results.

Given the time and computational constraints, we did not retrain our SVM re-ranker and NMT-based GEC system on the NUCLE data or tune them for $F_{0.5}$ on the CoNLL-2014 test set, making our results incomparable to those trained and optimised for NUCLE. However, our CLC-trained systems are still able to achieve competitive $F_{0.5}$ scores on the CoNLL-2014 test set without retraining. As our aim was to examine model generalisation and develop robust systems that are less likely to need retraining or tuning for new datasets or GEC tasks, we conclude that our GEC systems generalise well and can be used as generic systems across different tasks.

We also notice that almost all published GEC systems were trained using different datasets (e.g. NUCLE, CLC, Lang-8, WikEd Error Corpus, CommonCrawl or Web1T[1]) and optimised for different metrics (e.g. BLEU, M^2 or I-measure) on different test sets. It is necessary to compare all the systems under the same setting before we can draw any conclusion about the best GEC system and the true state-of-the-art performance. Thus, we propose a *closed* track error correction shared task, where participating teams are constrained to use only the provided training data, so that comparisons will be more likely focused on the methods, rather than the training data used.

Meanwhile, we strongly believe that a more representative test set is needed in order to better evaluate GEC system performance. Most published research has only reported system performance on the CoNLL-2014 test set. However, we argue

[1] `https://catalog.ldc.upenn.edu/ldc2006t13`

that it is not a representative test set of learner writing. The CoNLL-2014 test set consists of 50 essays written by 25 NUS students in response to two prompts. Similar to NUCLE, which contains essays produced by undergraduate students at NUS, the CoNLL-2014 test set is more likely to cover errors made by ESL learners whose L1s are Asian Languages (e.g. Chinese, Korean or Japanese) but less likely to contain errors from learners with other L1s. In addition, the topics discussed in the CoNLL-2014 test set are very limited, as essays collected in the test set were written in response to two prompts. Therefore, prompt-specific models are very likely to perform well on the test set. We also expect frequent repetition of some learner errors in the CoNLL-2014 test set given the fact that only 25 learners were recruited to write essays for the test data. The sample size might be too small to construct a representative test set. Compared with the CoNLL-2014 test set, the FCE dataset, which covers a wide variety of L1s and topics, seems to be a better option. This is why we decided to use the FCE dataset in our experiments in Chapter 4 and 5. However, the FCE dataset has its own limitations, as it only contains essays written by learners at an upper-intermediate level. Therefore, more efforts should be devoted to constructing a more representative test set of learner writing, on which different GEC systems should be tested.

During this work, there is an on-going discussion on how to evaluate GEC systems, and several methods have been proposed. So far there is no universally agreed evaluation measure for GEC, and the choice of a metric mainly depends on the application and research goals. For example, we used $F_{0.5}$ calculated by the M^2 scorer in Chapter 3 for participating in the CoNLL-2014 shared task, and the I-measure in Chapter 4 and 5 when we focussed on the improvement of the original text. It seems that most work on GEC has only reported $F_{0.5}$ lately, although as argued in this thesis, an increase in F-score in conjunction with the M^2 scorer does not necessarily mean a reduction in the actual error rate. Even when the increase in $F_{0.5}$ for recently published systems looks encouraging, we still do not know whether they will produce better corrections. Before we can agree on the best evaluation measure, we encourage future work to report system results using multiple evaluation metrics for better comparisons.

Evaluating system performance for each error type is very useful. As the error type information in all-errors GEC systems is missing, it comes as no surprise that there are very few published results. The type estimation strategy used in this thesis relied heavily on the heuristic rules extracted from NUCLE. Since datasets often use different annotation schemes, it would fail to generalise to new datasets so future work should continue to explore better ways to evaluate system performance by error type.

Finally, we would like to see the techniques developed in this thesis facilitate the development of GEC, as well as being used in real-world applications, like proofreading tools or educational software. We believe GEC techniques can help make language learning more accessible and interactive than ever before.

APPENDIX A

NUCLE ERROR CODES

Vt	Verb Tense
Vm	Verb modal
V0	Missing verb
Vform	Verb form
SVA	Subject-verb-agreement
ArtOrDet	Article or Determiner
Nn	Noun Number
Npos	Noun possessive
Pform	Pronoun form
Pref	Pronoun reference
Wcip	Wrong collocation/idiom/preposition
Wa	Acronyms
Wform	Word form
Wtone	Tone
Srun	Runons, comma splice
Smod	Dangling modifier
Spar	Parallelism
Sfrag	Fragment
Ssub	Subordinate clause
WOinc	Incorrect sentence form
WOadv	Adverb/adjective position
Trans	Link words/phrases
Mec	Punctuation, capitalization, spelling, typos
Rloc	Local redundancy
Cit	Citation
Others	Other errors
Um	Unclear meaning

APPENDIX B

CLC ERROR TAXONOMY

The letters which appear as the first letter of a bipartite error code indicate the type of error:

F	Form
M	Missing
R	Replace
U	Unnecessary
I	Inflection
D	Derivation
AG	Agreement
C	Countability

The letters which appear as the second letter of a bipartite error code indicate the POS that the error affects:

A	Pronoun (anaphora)
N	Noun
V	Verb
J	Adjective
T	Preposition
D	Determiner
C	Conjunction
Q	Quantifier
Y	Adverb
P	Punctuation

AG	Agreement error	M	Missing error
AGA	Anaphora agreement error	MA	Missing anaphor
AGD	Determiner agreement error	MC	Missing link word
AGN	Noun agreement error	MD	Missing determiner
AGV	Verb agreement error	MJ	Missing adjective
AGQ	Quantifier agreement error	MN	Missing noun
AS	Agreement structure error	MP	Missing punctuation
C	Countability error	MQ	Missing quantifier
CD	Wrong determiner because of noun countability	MT	Missing preposition
CE	Complex error	MV	Missing verb
CL	Collocation or tautology error	MY	Missing adverb
CN	Countability of noun error	NE	No error
CQ	Wrong quantifier because of noun countability	R	replace error
DA	Derivation of anaphor error	RA	Replace anaphor
DC	Derivation of link word error	RC	Replace link word
DD	Derivation of determiner error	RD	Replace determiner
DI	Incorrect determiner inflection	RJ	Replace adjective
DJ	Derivation of adjective error	RN	Replace noun
DN	Derivation of noun error	RP	Replace punctuation
DQ	Derivation of quantifier error	RQ	Replace quantifier
DT	Derivation of preposition error	RT	Replace preposition
DV	Derivation of verb error	RV	Replace verb
DY	Derivation of adverb error	RY	Replace adverb
FA	Wrong anaphor form	S	Spelling error
FC	Wrong link word form	SA	Spelling American
FD	Incorrect determiner form	SX	Spelling confusion
FJ	Wrong adjective form	TV	Incorrect tense of verb
FN	Wrong noun form	U	Unnecessary error
FQ	Wrong quantifier form	UA	Unnecessary anaphor
FT	Wrong preposition form	UC	Unnecessary link word
FV	Wrong verb form	UD	Unnecessary determiner
FY	Wrong adverb form	UJ	Unnecessary adjective
IA	Incorrect anaphor inflection	UN	Unnecessary noun
ID	Idiom wrong	UP	Unnecessary punctuation
IJ	Incorrect adjective inflection	UQ	Unnecessary quantifier
IN	Incorrect noun inflection	UT	Unnecessary preposition
IQ	Incorrect quantifier inflection	UV	Unnecessary verb
IV	Incorrect verb inflection	UY	Unnecessary adverb
IY	Incorrect adverb inflection	W	Word order error
L	Inappropriate register	X	Incorrect negative formation

Table B.1: CLC error taxonomy

Bibliography

Andersen, Ø. E., Yannakoudakis, H., Barker, F., and Parish, T. Developing and testing a self-assessment and tutoring system. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 32–41, Atlanta, Georgia, USA, June 2013.

Axelrod, A., He, X., and Gao, J. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK, July 2011.

Bahdanau, D., Cho, K., and Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA, May 2015.

Banerjee, S. and Lavie, A. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, USA, June 2005.

Bengio, Y., Simard, P., and Frasconi, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Bird, S., Loper, E., and Klein, E. *Natural language processing with Python*. O'Reilly Media Inc., 2009.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September 2015.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867, Prague, Czech Republic, June 2007.

Briscoe, T., Carroll, J., and Watson, R. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia, July 2006.

Brockett, C., Dolan, W. B., and Gamon, M. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 249–256, Sydney, Australia, July 2006.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Bryant, C. and Ng, H. T. 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*, pages 697–707, Beijing, China, July 2015.

Buck, C., Heafield, K., and van Ooyen, B. N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 3579–3584, Reykjavik, Iceland, May 2014.

Bustamante, F. R. and León, F. S. GramCheck: a grammar and style checker. In *Proceedings of the 16th International Conference on Computational Linguistic*, pages 175–181, Copenhagen, Denmark, August 1996.

Chen, S. F. and Goodman, J. An empirical study of smoothing techniques for language modeling. Technical report, Harvard University, USA, 1998.

Cherry, C. and Foster, G. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June 2012.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar, October 2014.

Chodorow, M., Tetreault, J. R., and Han, N.-R. Detection of grammatical errors involving prepositions. In *Proceeding of the 4th ACL-SIGSEM Workshop on Prepositions*, pages 25–30, Prague, Czech Republic, June 2007.

Chodorow, M., Dickinson, M., Israel, R., and Tetreault, J. Problems in Evaluating Grammatical Error Detection Systems. In *Proceedings of 24th International Conference on Computational Linguistics*, pages 611–628, Mumbai, India, December 2012.

Chollampatt, S., Taghipour, K., and Ng, H. T. Neural Network Translation Models for Grammatical Error Correction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2768–2774, New York City, New York, USA, July 2016.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *Proceedings of the Deep Learning and Representation Learning Workshop*, Montréal, Canada, December 2014.

Church, K. W. and Mercer, R. L. Introduction to the Special Issue on Computational Linguistics Using Large Corpora. *Computational Linguistics*, 19(1):1–24, 1993.

Collins, M. and Duffy, N. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Philadelphia, Pennsylvania, USA, July 2002.

Collins, M. and Koo, T. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1), 2005.

Collobert, R. Deep Learning for Efficient Discriminative Parsing. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 224–232, April 2011.

Connors, R. J. and Lunsford, A. A. Frequency of Formal Errors in Current College Writing, or Ma and Pa Kettle Do Research. *College Composition and Communication*, 39(4):395–409, 1988.

Costa-Jussà, M. R. and Fonollosa, J. A. R. Character-based Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 357–361, Berlin, Germany, August 2016.

Dahlmeier, D. and Ng, H. T. 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*, pages 568–578, Jeju Island, Korea, July 2012a.

Dahlmeier, D. and Ng, H. T. 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*, pages 568—-572, Montréal, Canada, June 2012b.

Dahlmeier, D., Ng, H. T., and Ng, E. J. F. NUS at the HOO 2012 Shared Task. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 216–224, Montréal, Canada, 2012.

Dahlmeier, D., Ng, H. T., and Wu, S. M. Building a large annotated corpus of learner english: the NUS Corpus of Learner English. In *Proceedings of the 8th*

Workshop on Innovative Use of NLP for Building Educational Applications, pages 22–31, Atlanta, Georgia, USA, June 2013.

Dale, R. and Kilgarriff, A. Helping Our Own: the HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242—-249, Nancy, France, September 2011.

Dale, R., Anisimoff, I., and Narroway, G. HOO 2012: a report on the preposition and determiner error correction shared task. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 54—-62, Montréal, Canada, June 2012.

Daudaravicius, V., Banchs, R. E., Volodina, E., and Napoles, C. A Report on the Automatic Evaluation of Scientific Writing Shared Task. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 53–62, San Diego, California, USA, June 2016.

De Felice, R. and Pulman, S. G. Automatically acquiring models of preposition use. In *Proceeding of the 4th ACL-SIGSEM Workshop on Prepositions*, pages 45–50, Prague, Czech Republic, June 2007.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, Maryland, USA, June 2014.

Donahue, S. Formal errors: Mainstream and ESL students. 2001. Presented at the 2001 Conference of the Two-Year College Association.

Durrani, N., Fraser, A., Schmid, H., Hoang, H., and Koehn, P. Can Markov Models Over Minimal Translation Units Help Phrase-Based SMT? In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 399–405, Sofia, Bulgaria, August 2013.

Farzi, S. and Faili, H. A swarm-inspired re-ranker system for statistical machine translation. *Computer Speech & Language*, 29:45–62, January 2015.

Federico, M., Bertoldi, N., and Cettolo, M. IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of the 9th Annual Conference of the International Speech Communication Association*, pages 1618–1621, Brisbane, Australia, September 2008.

Felice, M. and Briscoe, T. 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, pages 578–587, Denver, Colorado, USA, May–June 2015.

Felice, M. and Yuan, Z. To err is human, to correct is divine. *XRDS*, 21(1):22–27, October 2014a.

Felice, M. and Yuan, Z. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126, Gothenburg, Sweden, April 2014b.

Felice, M., Yuan, Z., Andersen, Ø. E., Yannakoudakis, H., and Kochmar, E. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland, USA, July 2014.

Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop*, 2008.

Gamon, M. High-order sequence modeling for language learner error detection. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 180–189, Portland, Oregon, USA, June 2011.

Gamon, M., Gao, J., Brockett, C., Klementiev, A., Dolan, W., Belenko, D., and Vanderwende, L. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 449–456, Hyderabad, India, January 2008.

Gao, J., Nguyen, P., Li, X., Thrasher, C., Li, M., and Wang, K. A comparative study of Bing Web N-gram language models for Web search and natural language processing. In *Web N-gram Workshop, Workshop of the 33rd Annual International ACM SIGIR Conference*, pages 16–21, Geneva, Switzerland, July 2010.

Glorot, X., Bordes, A., and Bengio, Y. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28 th International Conference on Machine Learning*, pages 513–520, Bellevue, Washington, USA, June 2011.

Goh, C.-L., Watanabe, T., Finch, A., and Sumita, E. Discriminative reranking for SMT using various global features. In *Proceedings of the 4th International Universal Communication Symposium*, pages 8–14, Beijing, China, 2010.

Good, I. J. The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, 40(3/4):237–264, 1953.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout Networks. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, June 2013.

Grundkiewicz, R. and Junczys-Dowmunt, M. The WikEd Error Corpus: A Corpus of Corrective Wikipedia Edits and its Application to Grammatical Error Correction. In *Advances in Natural Language Processing - Lecture Notes in Computer Science*, pages 478–490. Springer International Publishing, 2014.

Grundkiewicz, R., Junczys-Dowmunt, M., and Gillian, E. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 461–470, Lisbon, Portugal, September 2015.

Gui, S. and Yang, H., editors. 中国学习者语料库 *(Chinese) [Chinese Learner English Corpus]*. Shanghai Foreign Language Education Press, 2003.

Ha, T.-L., Niehues, J., and Waibel, A. Lexical Translation Model Using a Deep Neural Network Architecture. In *Proceedings of the 11th International Workshop on Spoken Language Translation*, pages 223–229, Lake Tahoe, California and Nevada, USA, December 2014.

Han, N.-R., Chodorow, M., and Leacock, C. Detecting errors in English article usage with a maximum entropy classifier trained on a large, diverse corpus. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1625–1628, Lisbon, Portugal, May 2004.

Heafield, K. and Lavie, A. CMU multi-engine machine translation for WMT 2010. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 301–306, Uppsala, Sweden, July 2010.

Heidorn, G. E., Jensen, K., Miller, L. A., Byrd, R. J., and Chodorow, M. The EPISTLE text-critiquing system. *IBM Systems Journal*, 21(3):305–326, 1982.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching Machines to Read and Comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc., 2015.

Hermet, M. and Désilets, A. Using first and second language models to correct preposition errors in second language authoring. In *Proceedings of the 4th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–72, Boulder, Colorado, June 2009.

Hildebrand, A. S. and Vogel, S. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas*, Hawaii, USA, October 2008.

Hoang, D. T., Chollampatt, S., and Ng, H. T. Exploiting N-Best Hypotheses to Improve an SMT Approach to Grammatical Error Correction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2803–2809, New York City, New York, USA, July 2016.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

Huang, Z., Harper, M. P., and Wang, W. Mandarin part-of-speech tagging and discriminative reranking. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1093–1102, Prague, Czech Republic, June 2007.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1–10, Beijing, China, July 2015a.

Jean, S., Firat, O., Cho, K., Memisevic, R., and Bengio, Y. Montreal Neural Machine Translation Systems for WMT15. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal, September 2015b.

Jeffreys, H. *Theory of probability*. Clarendon Press, Oxford, third edition, 1961.

Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, Edmonton, Canada, July 2002.

Joachims, T. Training linear SVMs in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–226, Philadelphia, Pennsylvania, USA, August 2006.

Johnson, W. E. Probability: The deductive and inductive problems. *Mind*, 41(164): 409–423, October 1932.

Junczys-Dowmunt, M. and Grundkiewicz, R. 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 18th Conference on Computational Natural Language Learning: Shared Task*, pages 25–33, Baltimore, Maryland, USA, June 2014.

Junczys-Dowmunt, M. and Grundkiewicz, R. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. *arXiv*, 1605.06353, 2016.

Kalchbrenner, N. and Blunsom, P. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013.

Knight, K. and Chander, I. Automated postediting of documents. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 865–872, Seattle, Washington, USA, August 1994.

Kochmar, E. Error Detection in Content Word Combinations. Technical report, University of Cambridge, UK, May 2016.

Koehn, P. *Statistical Machine Translation*. Cambridge University Press, 2010.

Koehn, P., Och, F. J., and Marcu, D. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Edmonton, Canada, May-June 2003.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180, Prague, Czech Republic, June 2007.

Kågebäck, M., Mogren, O., Tahmasebi, N., and Dubhashi, D. Extractive Summarization using Continuous Vector Space Models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality*, pages 31–39, Gothenburg, Sweden, April 2014.

Kukich, K. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992.

Kumar, S. and Byrne, W. Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 169–176, Boston, Massachusetts, USA, May 2004.

Kunchukuttan, A., Chaudhury, S., and Bhattacharyya, P. Tuning a Grammar Correction System for Increased Precision. In *Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task*, pages 60–64, Baltimore, Maryland, USA, June 2014.

Lado, R. *Linguistics Across Cultures: Applied Linguistics for Language Teachers*. University of Michigan Press, 1957.

Laplace, P.-S. *A philosophical essay on probabilities*. John Wiley & Sons, 1825. Translated from the sixth French edition by Frederick Wilson Truscott and Frederick Lincoln Emory, first edition, 1902.

Leacock, C., Chodorow, M., Gamon, M., and Tetreault, J. *Automated Grammatical Error Detection for Language Learners*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, second edition, 2014.

Lee, J. and Seneff, S. An analysis of grammatical errors in non-native speech in English. In *Proceedings of the 2008 IEEE Workshop on Spoken Language Technology*, pages 89–92, Goa, India, December 2008.

Lee, L.-H., Lin, B.-L., Yu, L.-C., and Tseng, Y.-H. The NTNU-YZU System in the AESW Shared Task: Automated Evaluation of Scientific Writing Using a Convolutional Neural Network. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 122–129, San Diego, California, USA, June 2016.

Levenshtein, V. I. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

Li, J. and Jurafsky, D. Mutual Information and Diverse Decoding Improve Neural Machine Translation. *arXiv*, 1601.00372, 2016.

Lidstone, G. J. Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192, 1920.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. Character-based Neural Machine Translation. *arXiv*, 1511.04586, 2015.

Luong, T., Pham, H., and Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015a.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. Addressing the Rare Word Problem in Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 11–19, Beijing, China, July 2015b.

MacDonald, N. H., Frase, L. T., Gingrich, P. S., and Keenan, S. A. The Writer's Workbench: Computer aids for text analysis. *IEEE Transactions on Communications*, 30(1), 1982.

Madnani, N., Tetreault, J., and Chodorow, M. Exploring grammatical error correction with not-so-crummy machine translation. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 44–53, Montréal, Canada, June 2012.

Meng, F., Lu, Z., Wang, M., Li, H., Jiang, W., and Liu, Q. Encoding Source Language with Convolutional Neural Network for Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 20–30, Beijing, China, July 2015.

Mikolov, T. and Zweig, G. Context Dependent Recurrent Neural Network Language Model. In *Proceedings of the 2012 IEEE Workshop on Spoken Language Technology*, pages 234–239, December 2012.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 1st International Conference on Learning Representations (Workshop Track)*, Scottsdale, Arizona, USA, May 2013.

Mitamura, T., Nyberg, E. H., and Carbonell, J. G. An Efficient Interlingua Translation System for Multi-lingual Document Production. In *Proceedings of Machine Translation Summit III*, Washington D.C, USA, July 1991.

Mizumoto, T. and Matsumoto, Y. Discriminative Reranking for Grammatical Error Correction with Statistical Machine Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1133–1138, San Diego, California, USA, June 2016.

Mizumoto, T., Komachi, M., Nagata, M., and Matsumoto, Y. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand, November 2011.

Mizumoto, T., Hayashibe, Y., Komachi, M., Nagata, M., and ji Matsumoto, Y. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of 24th International Conference on Computational Linguistics*, pages 863–872, Mumbai, India, December 2012.

Mnih, A. and Hinton, G. Three New Graphical Models for Statistical Language Modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648, June 2007.

Nagao, M. A framework of a mechanical translation between Japanese and English by analogy principle. In *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, Lyon, France, 1984.

Napoles, C., Sakaguchi, K., Post, M., and Tetreault, J. 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*, pages 588–593, Beijing, China, July 2015.

Napoles, C., Cahill, A., and Madnani, N. The Effect of Multiple Grammatical Errors on Processing Non-Native Writing. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–11, San Diego, California, USA, June 2016a.

Napoles, C., Sakaguchi, K., Post, M., and Tetreault, J. GLEU Without Tuning. *arXiv*, 1605.02592, 2016b.

Neubig, G., Watanabe, T., Sumita, E., Mori, S., and Kawahara, T. An unsupervised model for joint phrase alignment and extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 632–641, Portland, Oregon, USA, June 2011.

Neubig, G., Morishita, M., and Nakamura, S. Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation*, pages 35–41, Kyoto, Japan, October 2015.

Ney, H. and Essen, U. On smoothing techniques for bigram-based natural language modelling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 825–829, 1991.

Ney, H., Essen, U., and Kneser, R. On structuring probabilistic dependences in stochastic language modeling. *Computer, Speech, and Language*, 8:1–38, 1994.

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C., and Tetreault, J. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August 2013.

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., and Bryant, C. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, USA, June 2014.

Nicholls, D. The Cambridge Learner Corpus - error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 Conference*, pages 572–581, 2003.

Och, F. J. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July 2003.

Och, F. J. and Ney, H. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 161–168, Boston, Massachusetts, USA, May 2004.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, Atlanta, Georgia, USA, June 2013.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. How to Construct Deep Recurrent Neural Networks. In *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, April 2014.

Rei, M. and Yannakoudakis, H. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1181–1191, Berlin, Germany, August 2016.

Richardson, S. D. and Braden-Harder, L. C. The experience of developing a large-scale natural language text processing system: Critique. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 195–202, Austin, Texas, USA, February 1988.

Roark, B., Liu, Y., Harper, M., Stewart, R., Lease, M., Snover, M., Shafran, I., Dorr, B., Hale, J., Krasnyanskaya, A., and Yung, L. Reranking for sentence boundary detection in conversational speech. In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, May 2006.

Rozovskaya, A. and Roth, D. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 924–933, Portland, Oregon, USA, June 2011.

Rozovskaya, A. and Roth, D. Joint learning and inference for grammatical error correction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 791–802, Seattle, Washington, USA, October 2013.

Rozovskaya, A., Chang, K.-W., Sammons, M., and Roth, D. The University of Illinois system in the CoNLL-2013 shared task. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 13–19, Sofia, Bulgaria, August 2013.

Rozovskaya, A., Chang, K.-W., Sammons, M., Roth, D., and Habash, N. The Illinois-Columbia System in the CoNLL-2014 Shared Task. In *Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task*, pages 34–42, Baltimore, Maryland, USA, June 2014a.

Rozovskaya, A., Roth, D., and Srikumar, V. Correcting Grammatical Verb Errors. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 358–367, Gothenburg, Sweden, April 2014b.

Sakaguchi, K., Napoles, C., Post, M., and Tetreault, J. Reassessing the Goals of Grammatical Error Correction: Fluency Instead of Grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182, 2016.

Sato, S. and Nagao, M. Toward memory-based translation. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 247–252, Helsinki, Finland, August 1990.

Schmaltz, A., Kim, Y., Rush, A. M., and Shieber, S. 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*, pages 242–251, San Diego, California, USA, June 2016.

Schuster, M. and Paliwal, K. K. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, November 1997.

Sennrich, R., Haddow, B., and Birch, A. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany, August 2016.

Shannon, C. E. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Shen, L., Sarkar, A., and Josef Och, F. Discriminative reranking for machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 177–184, Boston, Massachusetts, USA, May 2004.

Simard, M., Goutte, C., and Isabelle, P. Statistical phrase-based post-editing. In *Proceedings of NAACL HLT*, pages 508–515, Rochester, NY, USA, April 2007.

Snover, M., Madnani, N., Dorr, B. J., and Schwartz, R. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of the 4th Workshop on Statistical Machine Translation*, pages 259–268, Athens, Greece, March 2009.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK, July 2011.

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

Sun, C., Jin, X., Lin, L., Zhao, Y., and Wang, X. Convolutional Neural Networks for Correcting English Article Errors. In *Natural Language Processing and Chinese Computing*, pages 102–110. Springer International Publishing, 2015.

Suresh, B. Inclusion of large input corpora in statistical machine translation. Technical report, Stanford University, USA, 2010.

Susanto, H. R., Phandi, P., and Ng, T. H. System combination for grammatical error correction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 951–962, Doha, Qatar, October 2014.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to Sequence Learning with Neural Networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

Tetreault, J. R. and Chodorow, M. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 865–872, Manchester, August 2008.

Ueffing, N. and Ney, H. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40, 2007.

van Rijsbergen, C. *Information Retrieval*. Butterworth-Heinemann, Newton, Massachusetts, USA, second edition, 1979.

Vapnik, V. N. *The nature of statistical learning theory*. Springer, New York City, New York, USA, 1995.

Vogel, S., Ney, H., and Tillmann, C. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark, August 1996.

Williams, C. The Cambridge Learner Corpus for researchers on the English Profile Project. Technical report, University of Cambridge ESOL Examinations, UK, 2008.

Wu, D. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora . *Computational Linguistics*, 23(3):377–403, 1997.

Wu, Y. and Ng, H. T. Grammatical error correction using integer linear programming. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1456–1465, Sofia, Bulgaria, August 2013.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. Neural Language Correction with Character-Based Attention. *arXiv*, 1603.09727, 2016.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048–2057, Lille, France, July 2015.

Xue, H. and Hwa, R. Redundancy Detection in ESL Writings. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 683–691, Gothenburg, Sweden, April 2014.

Yannakoudakis, H., Briscoe, T., and Medlock, B. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June 2011.

Yuan, Z. Error Detection/Correction of Chinese ESL Learners using SMT. Master's thesis, University of Cambridge, UK, June 2013.

Yuan, Z. and Briscoe, T. 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*, pages 380–386, San Diego, California, USA, June 2016.

Yuan, Z. and Felice, M. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria, August 2013.

Yuan, Z., Briscoe, T., and Felice, M. Candidate re-ranking for SMT-based grammatical error correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 256–266, San Diego, California, USA, June 2016.

Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *arXiv*, 1212.5701, 2012.